DATA REDUCTION METHODS IN JUST-IN-TIME-LEARNING

by

ONUR CAN BOY

B.S., Chemical Engineering, Boğaziçi University, 2018

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Chemical Engineering
Boğaziçi University
2021

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my thesis supervisor Assoc. Prof. Burak Alakent for his guidance, continuous support, patience and giving me opportunity to discover and do research in machine learning.

I am also very grateful to Assoc. Prof. Erdem Günay and Asst. Prof. Betül Uralcan for accepting to be a member of the thesis committee, sparing their valuable time for reading and commenting on my thesis.

Very special thanks to Deniz, Ekin, Gülten for their true friendship in good times and bad times and always being my side. I would also like to thank my roommate Kuntay Yiğit for his friendship.

I would like to thank my mother Songül and my sister Bilgen for their patience, encouragement, understanding, their unwavering support and belief in me.

I wish to extend my special thanks to Çiko, Pamuk, Haydut, Çırak, Babil and Şimbilli for always being there whenever I need a friend and a laugh.

ABSTRACT

DATA REDUCTION METHODS IN JUST-IN-TIME-LEARNING

Industrial processes are monitored continuously to meet the standards dictated by the market and environmental regulations. There are many process variables like temperature, pressure, flow rate which can be easily measured online through mechanical sensors and lots of data can be stored thanks to the advancements in data storage technology. At the same time, there are variables that show the quality of a product, process safety or some other restricted chemical composition. These are called quality variables and they are measured less often due to requirement of a detailed laboratory analysis. Measuring quality variable ones in a shift is a weak link in process control and monitoring. Implementing a soft sensor is a very efficient way to predict quality variables through statistical learning methods. Traditional soft sensors are built in an offline manner and used for online prediction while it requires maintenance periodically as process shifts to another state. Just-in-time learning is an adaptive method in which a local model is built when a new sample is obtained, and the model is discarded after a prediction is made. JITL outperforms traditional methods in terms of efficiency and predictive ability. The prediction performance of a soft sensor is also affected by the quality of the training data stored in the data base. Data reduction methods are used to eliminate data that weaken prediction quality and to store meaningful data for increasing prediction performance and model efficiency. In this thesis, JITL models are trained with Lasso and least squares support vector regression and three different data reduction algorithms using four different data sets. It is shown that the effect of each data reduction method changes from data set to data set, and prediction accuracy of JITL using all data can be attained using a smaller training sets. Additionally, results show that prediction accuracy of nonlinear models trained by LSSVR outperforms that of Lasso.

ÖZET

ANINDA ÖĞRENİM VE VERİ AZALTMA YÖNTEMLERİ

Endüstriyel süreçler, çevresel mevzuatlar ve pazardaki rekabet koşullarının belirlediği standartları sağlamak amacıyla sürekli olarak izlenmektedirler. Sıcaklık, basınç, akış hızı gibi süreç değişkenleri mekanik sensörler yardımıyla kolaylıkla ölçülmektedir ve veri depolama teknolojilerinin hızla gelişmesi sayesinde büyük miktarda süreç verisi depolanabilmektedir. Bununla birlikte bir ürünün kalitesini, süreçlerin güvenliğini ya da kısıtlanmış bir kimyasal kompozisyonunu gösteren değişkenler de vardır ve bunlar kalite değişkeni olarak adlandırılırlar. Kalite değişkenleri detaylı laboratuvar analizleri gerektirdiği için süreç değişkenlerine nazaran çok daha az sıklıkta ölçülmektedirler. Bu nedenle kalite değişkenlerinin takibi süreç kontrolü ve izlenmesinde zayıf noktalardan biridir. Sanal sensör uygulamaları kalite değişkenlerini istatistiksel öğrenme metotları kullanarak tahmin etmenin en verimli yöntemidir. Geleneksel sanal sensörler çevrimdışı şekilde modellenip çevrimiçi tahminlemede kullanılırlar. Bu nedenle süreç farklı koşullara kaydıkça düzenli olarak bakım çalışmaları gerekmektedir. Anında öğrenim, yeni bir numune geldikçe lokal bir model oluşturan yeni koşullara adapte olabilme yeteneğine sahip bir yöntemdir. Bu yöntemde tahminleme yapıldıktan sonra model bir daha kullanılmamaktadır. Anında öğrenim, geleneksel yöntemlere göre daha verimli ve tahminleme beceresi daha yüksek sonuçlar ortaya koymaktadır. Bir sanal sensörün tahminleme kalitesi aynı zamanda veri tabanında depolanmış verilerin kalitesine de bağlıdır. Veri azaltma yöntemleri tahminleme kalitesini düşüren verileri elimine etmek ve daha anlamlı verileri depolamak için kullanılan yöntemlerdir. Bu tez çalışmasında, anında öğrenim modelleri Lasso regresyonu ve destek vektör regresyonu istatistiksel öğrenme metotları kullanılarak dört farklı veri setinde üç farklı veri azaltma metodu kullanılarak incelenmiştir. Sonuç olarak veri azaltma yöntemlerinin veriminin veri setinden veri setine değiştiği ve anında öğrenim kullanıldığında tahminleme kalitesinin daha az sayıda veri kullanılarak korunduğu gösterilmiştir. Bununla birlikte lineer olmayan modeller lineer modellere göre daha iyi sonuçlar göstermiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	Subset of $\boldsymbol{X}$
d_{PE}	Number of samples
$E\{.\}$	Expected value
e_k	Error term in LSSVR
$GK(x_i, x_j)$	Gaussian Kernel of the vectors x_i and x_j
$f(.)$	True function
$\hat{f}(.)$	Predicted function
I	Identity matrix
$K(\boldsymbol{x}, \boldsymbol{z})$	Kernel function of vectors $\boldsymbol{x}$ and $\boldsymbol{z}$
N	Number of Samples
p	Number of predictors
P_{DMI}	Threshold of DMI method
S	Cost function
s_j	Standard deviation of the jth variable
$sim(x_i, x_j)$	Similarity between x_i and x_j
t	Tuning parameter in Lasso
t^0	Sum of coefficients of OLS estimate.
$Var\{.\}$	Variance
$\boldsymbol{w}$	Weight vector in SVM
X	Input matrix
$\boldsymbol{x}_i$	ith sample vector
$\mathbf{x}_j$	Array of input instances of jth regressor

x	Input variables
x_i	i^{th} input sample
x_{ij}	j^{th} regressor of i^{th} input sample
$\bar{\bar{\mathbf{x}}}_j$	Sample mean of j^{th} variable
$\mathbf{y}$	Output matrix
y	Output variable
$\hat{y}$	Predicted output
y_i	i^{th} observation of output variable
z_{ij}	Unit scaled input variables
α_i	Lagrangian multipliers
$\widehat{\boldsymbol{\beta}}$	Array of predicted coefficients
β	True coefficients
$\hat{\beta}$	Predicted coefficient
$\hat{\beta}^0$	OLS estimate
ϵ	Natural error
λ	Penalized regression parameter
ξ	Slack variable
$\varphi_i(.)$	Mapping to high dimensional input space
$\Delta_m(\boldsymbol{x}, \boldsymbol{z})$	Distance between vectors $\boldsymbol{x}$ and $\boldsymbol{z}$
γ	Regularization parameter in LSSVR
θ	Threshold in data reduction methods
σ	Kernel parameter in LSSVR
$\varnothing$	Null set

LIST OF ACRONYMS/ABBREVIATIONS

AD	Average Distance
ADPE	Average Distance and Prediction Error
KRR	Kernel Ridge Regression
LASSO	Least Absolute Shrinkage and Selection Operator
LS	Least Squares
MSE	Mean Squared Error
MW	Moving Window
NIPALS	Non-linear Iterative Partial Least Squares
NN	Nearest Neighbor
PCA	Principle Component Analysis
PLS	Partial Least Squares
RAE	Relative Additional Error
RMSE	Root Mean Squared Error
RR	Ridge Regression
SSR	Sum of Squares of Residuals
SVM	Support Vector Machines
JITL	Just-in-time Learning
OLS	Ordinary Least Square

1. INTRODUCTION

Industries are facing many challenges in decision making due to complications arising from many constraints in processes. Product quality and prices should satisfy the demand, as markets are getting more competitive. Environmental regulations offer additional issues for industries. Industries have to keep standards up to date in accordance with the new regulations implemented by governments and international environmental agreements. Industries also have to obey safety regulations strictly to prevent possible accidents [1]. For all these reasons, chemical processes are monitored constantly to maintain product quality at desired levels, to render lower energy consumption and safe working conditions in accordance with the environmental standards. Some of the process variables, like temperature, pressure, flow rate, and tank levels, are tracked second by second, as most of the production facilities are well equipped with temperature, flow rate, pressure, and level sensors. Some other variables, often referred to as quality variables, are measured less often due to a variety of reasons [2]. For such cases, implementing mechanical sensors can be difficult, mainly due to harsh process conditions, and/or the already installed sensors are in need of frequent maintenance, which result high upkeep costs. Furthermore, analyzer sensors might not be affordable by the company, or more complex laboratory analyses might be needed. For these reasons, quality variables are measured less often compared to process variables, usually with significant time delays that can extend hours. Some measurements are performed only once in a shift, so only a few measurements can be obtained in a day. Since online monitoring of the key variables is essential for process control, and optimization, online prediction of quality variables is a solution to this problem.

In the light of improvements in chemometrics, statistical learning methods and data storage technologies, “soft sensors” are being continuously used in the last few decades, and their popularity seems to be increasing [3]. Soft sensors (also known as inferential sensors, or virtual sensors) are cheap alternatives to hardware devices. They are widely used in many industries such as refineries, wastewater treatment, paper industry, cement plants, food processing, power plants, and other chemical-related plats. Soft sensors are generally categorized into two groups: model-driven soft sensors and data-driven soft sensors [4].

Model-driven soft sensors are based on first principle models, also known as white-box models, and they include the explicit thermodynamic and kinetic models, as well as mass and energy balances. These models are quite complex, hard to build, and often include many non-linearities, making it difficult to obtain solutions; even so, model might not work well. Data-driven soft sensors are built with process data through statistical learning algorithms. In data-driven soft sensors, one does not use explicit descriptions of physical phenomena and additional assumption that can lead to deviations from reality; instead the available process data is used in constructing models.

The predictive model is the main body of a soft sensor. A soft sensor may consist of a linear, or nonlinear model; there is no exact methodology for selecting the model type, and it is mostly up to the experience of the developer [4]. However, it is often suggested that a linear model (least squares, PLS, Ridge regression, LASSO) should be tried first, due to its simplicity and interpretability [2]. If the performance of the linear model is not high enough, then a convenient nonlinear model (support vector regression, nearest neighbor, fuzzy model) can be built after eliminating other possibilities that can lead to low performance.

The performance of a soft sensor also depends on the quality of the training data, for which there are numerous ways to improve. One of them is data size reduction (instance selection) [5]. Sometimes removing an instance from the training set does not affect the quality of prediction; on the contrary, this may even improve the prediction accuracy. In literature, the data size reduction problem is often worked for the classification problems. However, soft sensors are built mostly to predict continuous variables, due to the nature of chemical processes, and there are few works on instance selection methods for the regression problem [6]. Hence, this subject is open to development. There are generally two methods used for this purpose: The first one is removing instance according to distance measurements, and the other is turning the regression problem into a classification problem by assigning classes to the output variables (discretization). It is also essential that keeping training set relatively small can reduce runtimes significantly [7]. As size of the training set increases, it becomes more difficult to calculate the distance measurement between a query point and the past data, and to sort it.

In this thesis, it is aimed to design a soft sensor with high predictive accuracy, and simultaneously minimize the size of the training set. For this purpose, Lasso and least squares support vector regression (LSSVR) are used in prediction for four different data sets, and the suggested data reduction methods are compared with various methods from the literature. The thesis is organized as follows: Linear and non-linear statistical learning methods are explained in Section 2. In Section 3 and 4, general aspects of soft sensors, and adaptive soft sensing methods are discussed, respectively. Algorithms of the currently suggested data reduction methods are compared with those from the literature in Section 5, and numerical results are reported in Section 6.

2. STATISTICAL LEARNING METHODS

Statistical learning is a field used to determine a relation between input and output data, and make predictions using this relation. Utilization of statistical learning extends to the beginning of the nineteenth century. Linear regression and other linear models are widely used in many scientific fields throughout the years. While nonlinear models were devised in 1970s, they have gained popularity only since the mid-1980s with the increase of computational processing power [8].

In chemical processes, output variable is usually a quantitative variable, so the current study focuses exclusively on regression, rather than classification. Suppose that there is an output variable y, and input variables x, which consists of p different predictors. It is assumed that y is a function of x, and there is an error term ϵ with zero mean, and not correlated with predictors, i.e., measurement error, or unmeasured disturbances. This is also called irreducible error and it should be noted that y also depends on ϵ which has a zero mean and finite variance:

$$y = f(x) + \epsilon. \tag{2.1}$$

For most of the situations, the "true" relation (function f) between x and y is unknown. Hence, it is aimed to estimate this relation without knowing the exact model:

$$\hat{y} = \hat{f}(x). \tag{2.2}$$

Here, $\hat{y}$ and $\hat{f}$ denote that estimated values for y and f, respectively. Mean squared error of training samples (training MSE) is a most well-known method for evaluating performance of a model, quantitatively.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{f}(x_i)\right)^2. \tag{2.3}$$

On the other hand, to determine the generalization capacity of a model, one should focus on future predictions, in other words, test MSE. Increasing model complexity could result in a decrease in training MSE, but this can lead to a deterioration in a model's prediction performance, which is commonly known as over-fitting. Over-fitting is basically using several terms or degree of complexity that are not necessary [8]. Figure 2.1 shows an example of over-fitting. As training MSE (gray line) decreases, test MSE (red line) also decreases up to a point. After that point, test MSE starts to increase slightly, and when training MSE is getting closer to zero, test MSE becomes very large, and the overall trend looks like a U-shaped curve.

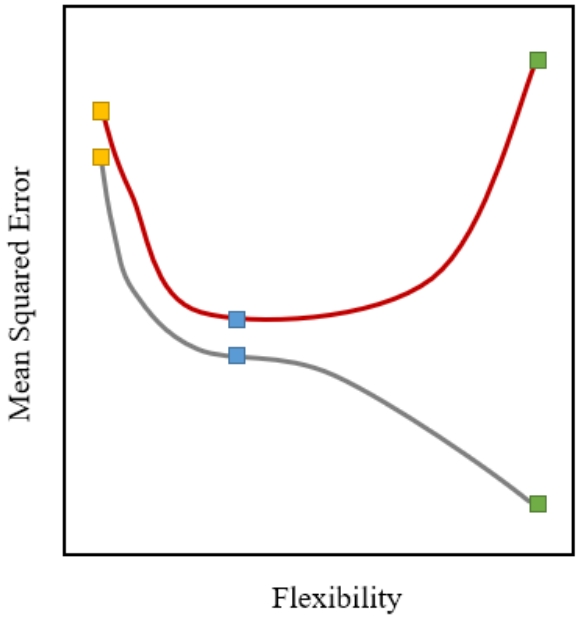


Figure 2.1. Training MSE (blue) and Test MSE (red).

Variance is the change in the estimated model upon using another training set. Generally, more complex models have more variance. Bias refers to the deviation from the true model. Too much simplification generally leads to high bias. For instance, a horizontal line fitted to data has a very large bias, since it is highly unlikely that a horizontal line reflects the true relation. As the model complexity changes, bias, and variance vary in different directions. Hence, an optimum point has to be achieved in order to minimize test MSE [9].

MSE can be decomposed into a relation between bias and variance under some assumptions. Suppose that a random sample of N elements $T_i = (x_i, y_i)$ for i $= 1, 2, \ldots, N$,

is to be used to derive a predictive function $\hat{f}(.)$, and it is desired to decrease the prediction error for a fixed observation x_0 with unknown true model response. This entails minimizing $\text{MSE}(\hat{y}_0)$ over infinite random samples of T_i, and infinite samples collected at x_0; hence expectation is taken over T_i and y_0. $\text{MSE}(\hat{y}_0)$ is decomposed in Equation 2.4 to 2.14.

$$MSE(\hat{y}_0) = E\left\{\left[y_0 - \hat{f}(x_0)\right]^2\right\}, \tag{2.4}$$

$$MSE(\hat{y}_0) = E\left\{\left[y_0 - E\{\hat{f}(x_0)\} + E\{\hat{f}(x_0)\} - \hat{f}(x_0)\right]^2\right\}, \tag{2.5}$$

$$\begin{aligned} MSE(\hat{y}_0) = E\left\{\left[y_0 - E\{\hat{f}(x_0)\}\right]^2\right\} + 2E\{[y_0 - \\ E\{\hat{f}(x_0)\}][E\{\hat{f}(x_0)\} - \hat{f}(x_0)]\} \; + \; E\left\{\left[E\{\hat{f}(x_0)\} - \hat{f}(x_0)\right]^2\right\}. \end{aligned} \tag{2.6}$$

There are three main terms in Equation 2.6 and these terms are examined separately in order to simplify derivation. It should be noted that $y_0 = f(x_0) + \epsilon$ in the beginning.

$$E\left\{\left[y_0 - E\{\hat{f}(x_0)\}\right]^2\right\} = \; E\left\{\left[f(x_0) - E\{\hat{f}(x_0)\} + \epsilon\right]^2\right\}, \tag{2.7}$$

$$\begin{aligned} E\left\{\left[f(x_0) - E\{\hat{f}(x_0)\} + \epsilon\right]^2\right\} = \; E\left\{\left[f(x_0) - E\{\hat{f}(x_0)\}\right]^2\right\} \; + \\ 2E\{\left[f(x_0) - E\{\hat{f}(x_0)\}\right][\epsilon]\} \; + \; E\{\epsilon^2\}. \end{aligned} \tag{2.8}$$

In Equation 2.8, $f(x_0) - E\{\hat{f}(x_0)\} = bias$, $f(x_0) - E\{\hat{f}(x_0)\}$ and ϵ are independent, $E\{\epsilon\} = 0$, and $E\{\epsilon^2\} = V\{\epsilon\}$. Using these assumptions Equation 2.8 is rearranged as follows:

$$E\left\{\left[f(x_0) - E\{\hat{f}(x_0)\} + \epsilon\right]^2\right\} = \; bias^2 \; + \; Var\{\epsilon\}. \tag{2.9}$$

The second term in Equation 2.6 is rewritten as following:

$$\begin{aligned} 2E\{[y_0 - E\{\hat{f}(x_0)\}][E\{\hat{f}(x_0)\} - \hat{f}(x_0)]\} \\ = 2E\{\left[f(x_0) - E\{\hat{f}(x_0)\} + \epsilon\right]\left[E\{\hat{f}(x_0)\} - \hat{f}(x_0)\right]\}. \end{aligned} \tag{2.10}$$

In Equation 2.10, ϵ is independent of $\hat{f}(x_0)$ and x_0. Since $E\{\epsilon\} = 0$, the terms which include ϵ are zero. Equation 2.10 can be rewritten without ϵ. Also, $f(x_0) - E\{\hat{f}(x_0)\} = bias$, and note that $bias$ is not a random variable, but a constant term.

$$2E\{[bias][E\{\hat{f}(x_0)\} - \hat{f}(x_0)]\} = 2[bias]\{[E\{\hat{f}(x_0)\} - \hat{f}(x_0)]\}, \quad (2.11)$$

$$E\{[E\{\hat{f}(x_0)\} - \hat{f}(x_0)]\} = E\{\hat{f}(x_0)\} - E\{\hat{f}(x_0)\} = 0. \quad (2.12)$$

The last term in Equation 2.8 is can be rewritten as variance:

$$E\left\{[E\{\hat{f}(x_0)\} - \hat{f}(x_0)]^2\right\} = Var\{\hat{f}(x_0)\}. \quad (2.13)$$

Equation 2.9, 2.12, and 2.13 are resulting forms of the three terms in Equation 2.6, respectively. The final form the derivation is given in Equation 2.14.

$$MSE(\hat{y}_0) = bias^2 + Var\{\epsilon\} + Var\{\hat{f}(x_0)\}. \quad (2.14)$$

It should be noted that $Var\{\epsilon\}$ is variance of irreducible error. As a conclusion, one can minimize $MSE(\hat{y}_0)$ by minimizing the total of model bias and model variance ($Var\{\hat{f}(x_0)\}$) terms.

2.1. Linear Regression

Linear regression is a technique to model a linear relationship between the response variable (outputs, dependent variables) and regressors (predictors, input variables, independent variables). If there is only one regressor, it is called simple linear regression, whereas linear regression with more than one regressor is called multiple linear regression [10]. The regression equation is basically an approximation of the true relation between the

response variable and regressors. Equation 2.15 indicates that y for the i^{th} observation has to be a linear function of β's and an irreducible random error ϵ, which is an independent variable with zero mean and finite variance.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i, \quad i = 1, 2, \ldots N. \tag{2.15}$$

Equation 2.15 consist of p regressors, $\mathbf{x}_j = \left[x_{1j}\ x_{2j}\ \ldots\ x_{Nj}\right]^{\boldsymbol{T}}, j = 1, 2, \ldots p$. Parameters $\beta_j, j = 1, 2, \ldots p$ are the regression coefficients and each β_j represents the expected change in y upon a unit change in $\mathbf{x}_j$, while other regressors are held constant. A simple linear regression model is shown in Equation 2.16 that describes the relation between the response variable and the regressor via a single straight line.

$$y_i = \beta_0 + \beta_1 x_{1i} + \epsilon_i, \quad i = 1, 2, \ldots N. \tag{2.16}$$

Least squares is the most well-known method used to estimate the parameters in linear regression. For ordinary least squares, $\hat{\beta}_0$ and $\hat{\beta}_1$ are determined so as to minimize the sum of squares of the difference between observed data and predicted values, named residuals. For *N* observations, sum of squares of residuals (SSR) is shown as follows:

$$\mathbf{S}(\beta_0, \beta_1) = \sum_{i=1}^{N} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{1i}\right)^2. \tag{2.17}$$

Least squares solution of multiple linear regression is very similar to simple linear regression. The only difference is that it is usually represented by matrix operations:

$$\mathbf{S}\left(\beta_0, \beta_1, \beta_1, \ldots, \beta_p\right) = \sum_{i=1}^{N} \left(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} \hat{\beta}_j x_{ij} \right)^2. \tag{2.18}$$

Taking the partial derivative of Equation 2.18 with respect to each regression parameter and equating to zero leads to p+1 equations. Solution of these equations yields:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}. \tag{2.19}$$

In the above equation, $\mathbf{X} = \left[\mathbf{1}_{N\times 1}\ \mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_p\right]_{N\times p}$, $\mathbf{y} = [y_1\ y_2\ \ldots\ y_N]^{\boldsymbol{T}}$, $\widehat{\boldsymbol{\beta}} = [\beta_0\ \beta_1\ \ldots, \beta_p]^{\boldsymbol{T}}$ and $\mathbf{1}_{N\times 1} = [1\ 1\ \ldots 1]^{\boldsymbol{T}}$**.** There is a one restriction for the unique solution of $\widehat{\boldsymbol{\beta}}$; $(\mathbf{X}'\mathbf{X})^{-1}$ must exist. This occurs when predictors are linearly independent. One of the properties of the least squares estimator $\widehat{\boldsymbol{\beta}}$ is that it is an unbiased estimator of $\boldsymbol{\beta}$ as previously mentioned:

$$E(\hat{\beta}) = E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}] = E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon})\}, \tag{2.20}$$

$$= E[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}\boldsymbol{\beta} + (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\varepsilon] = \boldsymbol{\beta}. \tag{2.21}$$

as $E(\boldsymbol{\varepsilon}) = 0$ and $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X} = \mathbf{I}$. The absolute magnitude of estimated coefficients, $\hat{\beta}$'s, are affected by the units of regressors so that it would be difficult to compare coefficients. Hence, it is beneficial to scale predictors and response variables. There are different standardization methods, but unit normal scaling is widely used in many statistical learning methods besides linear regression. Below, $\bar{\bar{\mathbf{x}}}_j$ and s_j refer to the sample mean and standard deviation of the j^{th} variable (column).

$$z_{ij} = \frac{x_{ij} - \bar{\bar{\mathbf{x}}}_j}{s_j}, i = 1, 2, \ldots . N; j = 1,2, \ldots p. \tag{2.22}$$

It should be noted that least-squares solutions lead to a singularity for the inverse of $\mathbf{X}'\mathbf{X}$**,** in the presence of linearly dependent predictors. As a result, the estimated coefficients may have unrealistically large magnitudes, and/or have wrong signs. Since there are tens of process variables measured in chemical processes, it is natural that they are highly correlated due to conservation laws and physical relations. Hence, multicollinearity is encountered very

often in industry. There are various methods to deal with multicollinearity, which will be discussed in the following sections.

2.2. Partial Least Squares

Partial least squares is a popular linear statistical learning method that was developed in the 1960s. It is a viable alternative to least-squares solution and principal component analysis with the capability of working with collinear, noisy, and many variables [11]. Although principal component regression (PCR) eliminates the collinearity problem, it can result in missing some information in the discarded principle components, and there might some noise in the remaining components used for regression. In addition to these, previous methods use the assumption that there is no measurement error, which is rarely seen in real life situations, while this is not a problem for PLS due to simultaneous modeling of relations among independent variables and dependent variables [12,13]. PLS is used for modeling large data structures in terms of chains of blocks, and it is aimed to find a way to relate **X** and **Y** data blocks [14]. High dimensional data is projected to a common structure space by regression-based methods. Non-linear iterative partial least squares (NIPALS) is a simple iterative method to obtain PLS components one by one, consisting of both external and inner relations of **X** and **Y** blocks.

2.3. Penalized Regression

Penalized regression methods have received significant attention in the last few decades. A large number of regressors often cause collinearity problems, which results in poor fitting and predictions. As well as prediction performance, one might desire to build a more interpretable model by choosing a smaller subset of predictors. This encourages people to pay attention to regression methods with different penalties to get good prediction performance. It is a useful technique for not only collinear predictors but also when the number of predictors is much larger than the number of observations, $N \ll p$ [15].

2.3.1. Ridge Regression

Ridge regression was first proposed in 1940s and has been widely used since. It basically consists of addition of a small positive number (λ) to the diagonal of the correlation matrix, as shown in Equations 2.23 and 2.24. This constant, while causing a small increase in bias, is used to reduce the variance significantly, and hopefully decrease the MSE [17].

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{Y}, \;\; \lambda > 0. \tag{2.24}$$

Ridge regression can be also expressed as Lagrangian:

$$min\left(\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{k=1}^{p}{\beta_k}^2\right). \tag{2.25}$$

Here, simulated data is used to demonstrate how ridge regression works. The response variable is determined using a single predictor, and seven additional predictors are obtained adding random noise to the single predictor and included in the regression model. Figure 2.2 shows the value of each eigenvalue and the minimum eigenvalue is around 0.1. The smallest eigenvalue indicates a near multi-collinear behavior, and is likely to make the regression parameter estimates unstable.

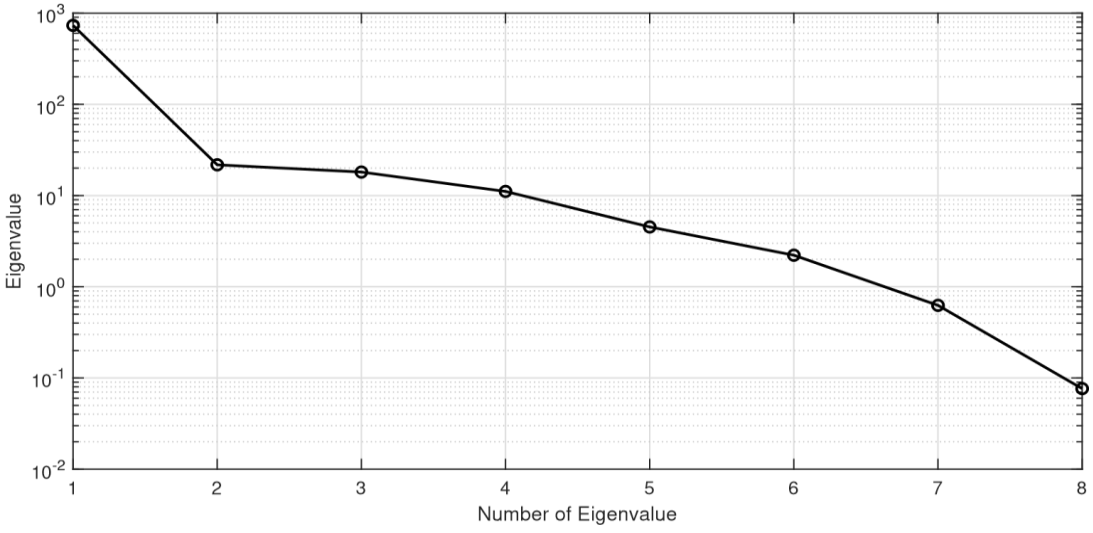


Figure 2.2. Eigenvalues of $\boldsymbol{X'X}$ matrix.

Adding a small λ in ridge regression increases the value of the smallest eigenvalue, and stabilizes the parameter estimates. Figure 2.3 shows how estimated coefficients change as λ increases. It should be noted that some of the coefficients change their signs.

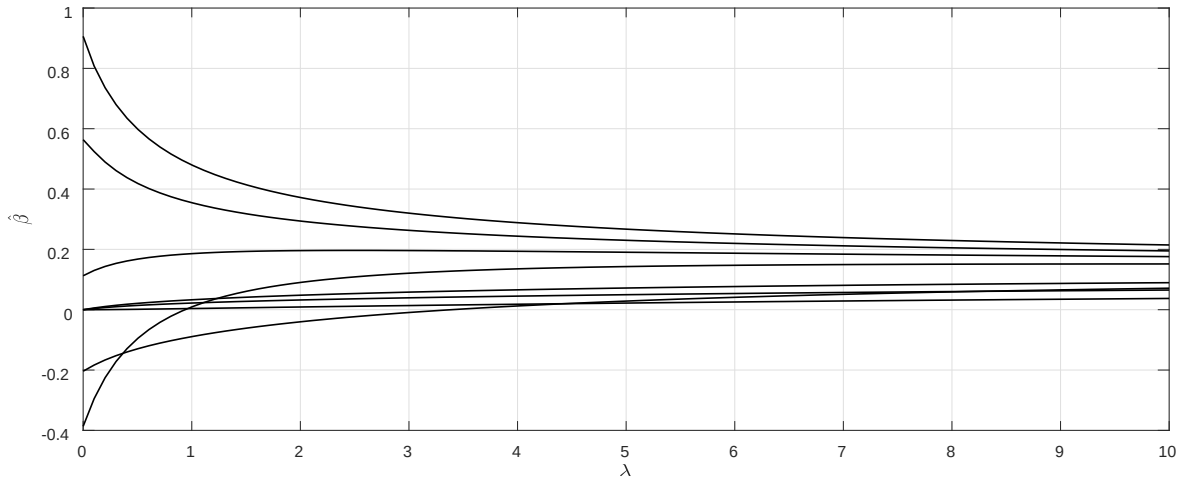


Figure 2.3. Variation of estimated coefficients with respect to λ.

2.3.2. Lasso

In Lasso (least absolute shrinkage and selection operator), the sum of squares of residuals is minimized subject to constraining the sum of the absolute values of the parameter estimates, that may result some of the estimated coefficient to be equal zero [18]. While advantage of shrinking regression parameter estimates is already exploited in ridge regression, setting some of the regression parameters to be equal zero, particularly in the presence of a large number of predictors, would be quite useful in terms of interpretable results. [19]. Equation 2.26 shows the loss function, and the inequality constraint is given in Equation 2.27.

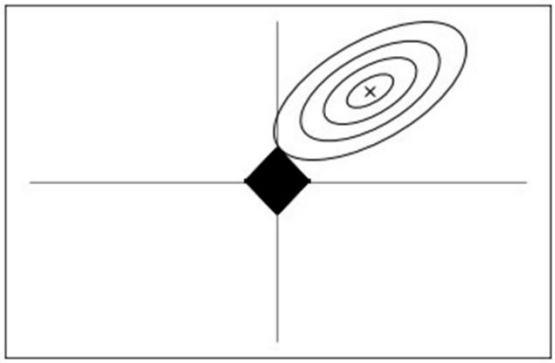
$$(\hat{\alpha}, \hat{\beta}) = min\left\{\sum_{i=1}^{N}\left(y_i - \alpha - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2\right\}, \tag{2.26}$$

$$\text{subject to} \quad \textstyle\sum_j \left|\beta_j\right| \leq t.$$

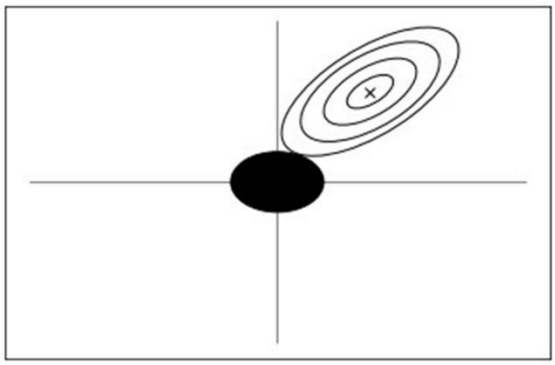
Here, the sum of the coefficients is constrained, and t is the tuning parameter. This is also a QP (quadratic programming) problem with an inequality constrain. If coefficients of

the LS estimates is $\hat{\beta}^0$, then sum of the LS estimates is $\sum_j |\hat{\beta}^0| = t^0$. When a $t < t^0$ is chosen as a constraint, coefficients are shrunk through zero.

Figure 2.4 shows the difference between ridge regression and lasso from a geometrical perspective. The center of ellipses represents the LS estimate in a two predictors space, and each ellipse represents various solutions with equal bias, with bias increasing as the solution is farther away from the center. An ellipse might touch the constrained area on any of the axis due to the shape of constrained area in Figure 2.4a whereas it is very difficult that an ellipse can touch the constrained area of ridge regression on the axis as shown in Figure 2.4b.



(a)



(b)

Figure 2.4. Geometric representation of shrinkage for (a) Lasso and (b) ridge regression.

Constraint of the Lasso is often written in a Lagrangian form as given in Equation 2.27. The constant, λ, determines the magnitude of shrinkage by multiplying penalty term. It is often found through a grid search that minimizes training error. The main difference from the Lagrangian of the ridge regression, which is shown in Equation 2.25, is the sum of

the absolute values of coefficients is used as a penalty term in loss function instead of the sum of squares of coefficients.

$$min\left(\sum_{i=1}^{N}\left(y_i-\sum_{j=1}^{N}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|\right). \tag{2.27}$$

2.4. Support Vector Machine

The support vector machine is a supervised statistical learning method introduced by Vapnik [20] in 1995. SVM is used for modelling of nonlinear patterns properly and separating different classes from each other in a most efficient way. Figure 2.5a shows how a hyperplane separates two classes in a two-dimensional space. There are infinitely many hyperplane options to do that. Vapnik states that the closest points to the hyperplane must satisfy the condition: $\mathbf{w}^{\mathrm{T}}\boldsymbol{x}+\mathrm{b}=1$, and aim to maximize the distance between the hyperplane and the nearest points to the hyperplane as shown in Figure 2.5b. The following formulation assumes a linear classifier to perform the classification task [21]:

$$y(\boldsymbol{x})=sig\,n(\boldsymbol{w}^T\boldsymbol{x}+b), \tag{2.28}$$

$$\begin{cases}\boldsymbol{w}^T\boldsymbol{x}_i+b\geq+1, & \text{if } y_i=+1\\ \boldsymbol{w}^T\boldsymbol{x}_i+b\leq-1, & \text{if } y_i=-1\end{cases}. \tag{2.29}$$

The two inequalities in the Equation 2.29 can be gathered in a single equation:

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)\geq 1. \tag{2.30}$$

The SVM formulation is done using convex optimization. Firstly, constrained optimization problem is defined. Then, the problem is transformed to Lagrangian form and problem is solved using Lagrange multipliers. These will be called support vectors.

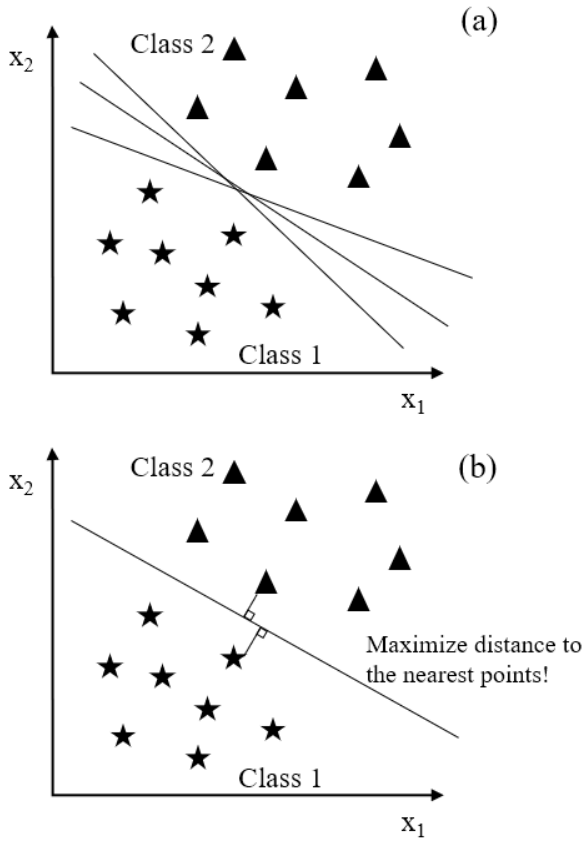


Figure 2.5. (a) The different options and (b) the one which maximize the distance to nearest points for separating hyperplane.

$$\min_{\boldsymbol{w},\boldsymbol{b}} J_{\mathbf{P}}(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}, \tag{2.31}$$

$$such\ that\ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 \text{ for } i = 1, 2, \ldots, N. \tag{2.32}$$

The Lagrangian form of the problem above is given in Equation 2.33 with Lagrangian multipliers $\alpha_i \geq 0$ for $i = 1, 2, .,N$:

$$\mathcal{L}(\boldsymbol{w}, b; \alpha) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} \alpha_i(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1). \tag{2.33}$$

However, this formulation is used for totally separable case. In real life, it is highly unlikely to encounter such cases. Hence, it is aimed to separate classes as much as possible. In order to do that, Vapnik [22] introduced slack variables into the formulation.

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i. \tag{2.34}$$

Then, optimization problem is revised as follows:

$$\begin{gathered} \min_{\boldsymbol{w},b} J_{\mathrm{P}}(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + c\sum_{i=1}^{N}\xi_i, \\ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1,2,\dots,N, \\ \xi_i \geq 0. \end{gathered} \tag{2.35}$$

In addition to classification problem, SVM can also be used for regression analysis. Vapnik's intensive cost function is used for this purpose [22]:

$$\begin{gathered} \min_{\boldsymbol{w},b} J_P(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}, \\ y_i - \boldsymbol{w}^T\boldsymbol{x}_i - b \leq \epsilon, i = 1,\dots,N \\ \boldsymbol{w}^T\boldsymbol{x}_i + b - y_i \leq \epsilon, i = 1,\dots,N \end{gathered} \tag{2.36}$$

2.5. Least Squares Support Vector Machine

LSSVM is a modification of SVM formulation. The motivation behind this is to simplify SVM formulation without losing its advantages in order to avoid QP. There are two main differences. The inequality constraint is changed with equality constraint so that the value 1 is proposed as a target variable rather than a threshold. The error term e_i is the equivalent of the slack variable ξ_i in the SVM formulation. The other aspect is that squared loss function is implemented with this error term, and this leads to some simplifications in solution. Again, the convex optimization is done for a classification problem at first, and it will be extended to regression analysis as in SVM [21]:

$$\begin{gathered} \min_{\boldsymbol{w},b} J_P(\boldsymbol{w}, e) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \gamma\frac{1}{2}\sum_{i=1}^{N} e_i^2 \\ y_i(\boldsymbol{w}^T\varphi_i(\boldsymbol{x}) + b) = 1 - e_i \end{gathered}. \tag{2.37}$$

The Lagrangian multipliers might be positive or negative because equality constraint is used. $\varphi_i(\boldsymbol{x})$ is mapping to high dimensional input space. The motivation behind this is that data might be separable in a higher dimensional space that can be even infinite

dimensional by a linear separating hyperplane. This application is also called kernel trick. There is a map $\varphi_i(\boldsymbol{x})$ for any symmetric and continuous function. Hence, $\varphi_i(\boldsymbol{x})$ does not have to be known explicitly, and kernel function can be expressed as the dot product [21]:

$$K(\boldsymbol{x}_i, \boldsymbol{z}) = \varphi_i(\boldsymbol{x}_i)^T \varphi_i(\boldsymbol{z}). \tag{2.38}$$

Widely used kernel functions are linear, polynomial and Gaussian kernels [23]:

$$K_{\text{linear}}(\boldsymbol{x}_i, \boldsymbol{z}) = \boldsymbol{x}^T \mathbf{z}, \tag{2.39}$$

$$K_{\text{polynomial}}(\boldsymbol{x}_i, \boldsymbol{z}) = (\boldsymbol{x}^T \mathbf{z} + 1)^d, \tag{2.40}$$

$$K_{Gaussian}(\boldsymbol{x}_i, \boldsymbol{z}) = exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{z}\|^2}{\sigma^2}\right). \tag{2.41}$$

The Lagrangian of the Equation 2.37:

$$\mathcal{L}(\boldsymbol{w}, b, e; \alpha) = J_p(\boldsymbol{w}, e) - \sum_{i=1}^{N} \alpha_i \left(y_i \left(\boldsymbol{w}^T \varphi_i(\boldsymbol{x}) + b\right) - 1 + e_i\right). \tag{2.42}$$

Solution is given in Equation 2.43 by solving Karush-Kuhn-Tucker system which is a set of equations that are necessary for optimality [21]. It is ensured that constraints are not violated, and gradient of the objective function is zero in any direction at the optimum points via KKT.

$$y(\boldsymbol{x}) = sign\left(\sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b\right). \tag{2.43}$$

The main difference of the solution is that there is no zero α_k, and each point is a support vector. The larger $|\alpha_k|$ means that this point in the training set is closer to decision boundary than other points but, more or less, all data points are used for building the model. Once may extent the classification to regression problem [21]:

$$\begin{aligned} &\min_{\boldsymbol{w},b} J_P(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \gamma \frac{1}{2}\sum_{i=1}^{N} e_i^2, \\ &y_i = \boldsymbol{w}^T \varphi_i(\boldsymbol{x}) + b + e_i. \end{aligned} \tag{2.44}$$

Lagrangian is constructed in an identical fashion that that in equation 2.42. The final LSSVM model for regression becomes:

$$y(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b. \tag{2.45}$$

where α_i and b are the solutions of the linear system and $K(\boldsymbol{x}, \boldsymbol{x}_i)$ is the kernel function.

2.6. Kernel Ridge Regression

Ridge regression is the simplest algorithm to be kernelized. Input space is transferred to higher dimensional space, which can be infinitely large, and a linear model is built on that space. If the kernel is non-linear, then the model built is also non-linear in the input space. Equation 2.24 and 2.25 show the solution of ridge regression [24,24]. The input data matrix is replaced with feature vector (Equation 2.46) and matrix inversion lemma is a trick that makes it possible to perform inverse application in the smallest possible dimensions as shown in Equations 2.47 and 2.48:

$$\boldsymbol{x} \rightarrow \varphi(\boldsymbol{x}), \tag{2.46}$$

$$w = [\varphi(\boldsymbol{X})\varphi(\boldsymbol{X})^T + \lambda \boldsymbol{I}]^{-1}\varphi(\boldsymbol{x})\boldsymbol{Y}, \tag{2.47}$$

$$w = \varphi(\boldsymbol{X})^T[\varphi(\boldsymbol{X})\varphi(\boldsymbol{X})^T + \lambda \boldsymbol{I}]^{-1}\boldsymbol{Y}. \tag{2.48}$$

The most important property of the kernel ridge regression is that we do not need to know feature vector, which might have infinitely many dimensions. For this, it is only needed to have a test point x_i:

$$\hat{y}(\boldsymbol{x_q}) = \varphi(\boldsymbol{x_q})^T w, \tag{2.49}$$

$$\hat{y}(\boldsymbol{x_q}) = \varphi(\boldsymbol{x_q})^T \varphi(\boldsymbol{X})^T[\varphi(\boldsymbol{X})\varphi(\boldsymbol{X})^T + \lambda \boldsymbol{I}]^{-1}\boldsymbol{Y}. \tag{2.50}$$

where $\boldsymbol{K} = \varphi(\boldsymbol{X})\varphi(\boldsymbol{X})^T$ and $\boldsymbol{\kappa}(\boldsymbol{x_q}) = \varphi(\boldsymbol{x_q})^T\varphi(\boldsymbol{X})^T$. Hence, Equation 2.50 becomes:

$$\hat{y}(\boldsymbol{x_q}) = \boldsymbol{\kappa}(\boldsymbol{x_q})[\boldsymbol{K} + \lambda I]^{-1}\boldsymbol{Y}. \tag{2.51}$$

Figure 2.6 shows a fitting on randomly generated non-linear data and the effect of σ when Gaussian radial basis function is used in Equation 2.41. As σ increases, smoothness of the fitted model increases at the cost of higher bias.

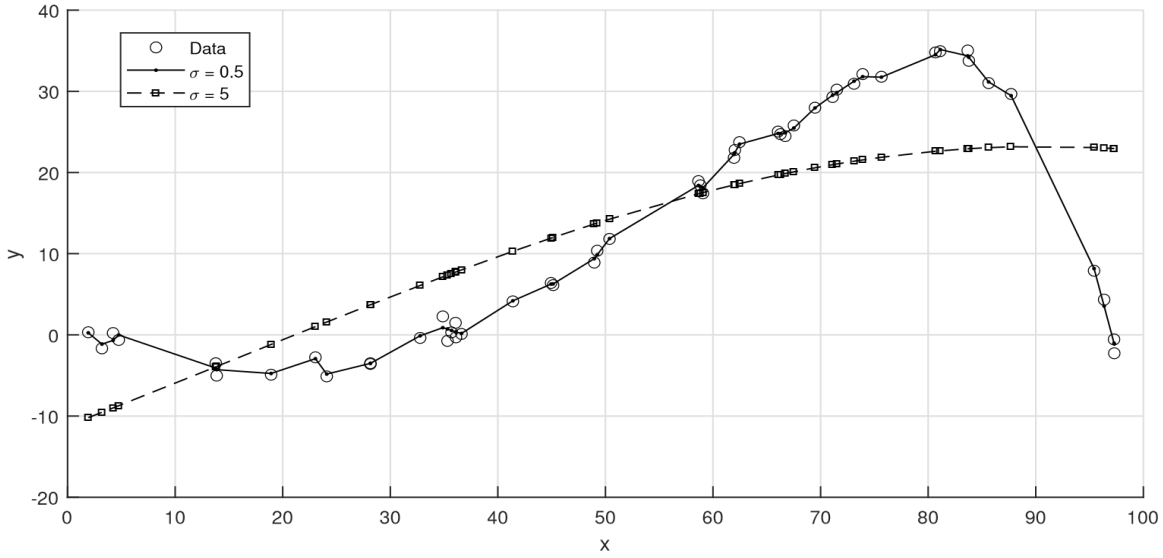


Figure 2.6. An example of KRR fit with Gaussian RBF and dependence on σ.

3. AN OVERVIEW OF SOFT SENSORS

In industrial processes, it is crucial to secure high product quality and process safety. Environmental regulations also force companies to put strict limitations on some key variables. Process monitoring is a widely adopted method to ensure aforementioned standards. Process monitoring can be grouped under three categories: knowledge-based methods, model-based methods and data-driven methods [25]. Knowledge-based method depends on the experience of the plant operators and engineers. It requires time and continuity of workers. Model-based methods (first-principle models) consist of fundamental physical and chemical relationships between process variables. Model-based methods give highly accurate results, however, it is difficult to build these models due to complex nature of the processes. For data-driven models, physical characteristics of a process and expert knowledge are not required to be known [26], hence data-driven methods (black-box models) have been getting more attention due to their practicality and low costs, and due to availability of large amount of process data.

Data-driven soft sensors, also known as virtual online analyzers, are inferential sensors that provide online predictions of difficult to measure variables using recorded process data in industrial plants. Soft sensors are cheap, efficient and reliable alternatives of expensive analyzers [27]. As previously mentioned, soft sensors are easier to build via statistical learning methods compared to building first-principle models. Soft sensors are being widely used for i) process control applications, such as advanced process control, model predictive control, operational planning and scheduling, and ii) offline operational assistance, such as diagnosis, simulations, and design, and iii) process monitoring for the last two decades [27].

Process data recorded in industry, generally, shows some important characteristics. Firstly, there is often assumed to be a linear relationship between the input variables and the quality variable due to stable operation conditions [26]. In other words, even if the process characteristics are nonlinear, linear models might work well around steady operating conditions. However, complex processes that can work under various operation modes show strong nonlinearity. There are also many measured variables, which lead to a high

dimensional data, and measurements obtained from successive mechanical sensors are highly correlated [28]. Process shifts are another issue that can lead to time-varying behavior of the process, i.e. similar input data might be related with different process modes [28]. Catalyst deactivation, aging of equipment, different raw materials, fouling in pipes, and seasonal changes are the main reason of process shifts. All these characteristics are the main challenges encountered while using data-driven models.

Before building soft sensor, the recorded data should be pre-processed. Raw data often consists of outliers, missing data and abnormal deviations from normal values. A good prediction performance cannot be achieved in the presence of these samples [29]. An outlier detection must be utilized, such as 3σ edit rule used for labeling samples that are more than three standard deviations away from the mean. Missing data is often related with mechanical sensor errors, or plant shut downs. Model-based interpolation can be used for non-systematic errors, but missing data blocks related with plant shut downs must be removed [30].

A previous survey with engineers showed that deterioration in prediction quality due to changes in process conditions is the most common problem related with soft sensors [31]. Hence, a soft sensor should be updated as the process conditions change in order to maintain its predictive ability. Different self-updating techniques (adaptive methods) are developed through years such as moving windows, recursive methods, time difference [32]. Each method has some advantages and disadvantages. Time difference is a simple and stable method, but over-fitting to a narrow operating range is one of its drawbacks. In moving window, a specific number of the latest obtained samples are used for training the model, but the appropriate window size is a crucial parameter. Just-in-time learning (also known as instance-based learning, lazy learning, model-on demand) is an alternative way to build adaptive soft sensors, which copes with nonlinearity and changes in process characteristics, while avoiding drawbacks of other adaptive methods [33]. In the following subsections, JITL, moving window and time difference modeling are discussed, respectively.

3.1. Just-in-time Learning

Just-in-time learning is an efficient and flexible online learning method. In JITL, a local model is built around a query point to cope with changes in the process conditions and strong

nonlinearity [31]. It is assumed that all data points are stored in the database, and model building is postponed until a prediction of output variable is required [33]. As Figure 3.1 shows the schematic representation of the JITL, a query point is obtained and relevant samples around the query point are selected from the training set. Unlike offline modelling, a local model is built with relevant samples. After the prediction is made, the built model is discarded [34].

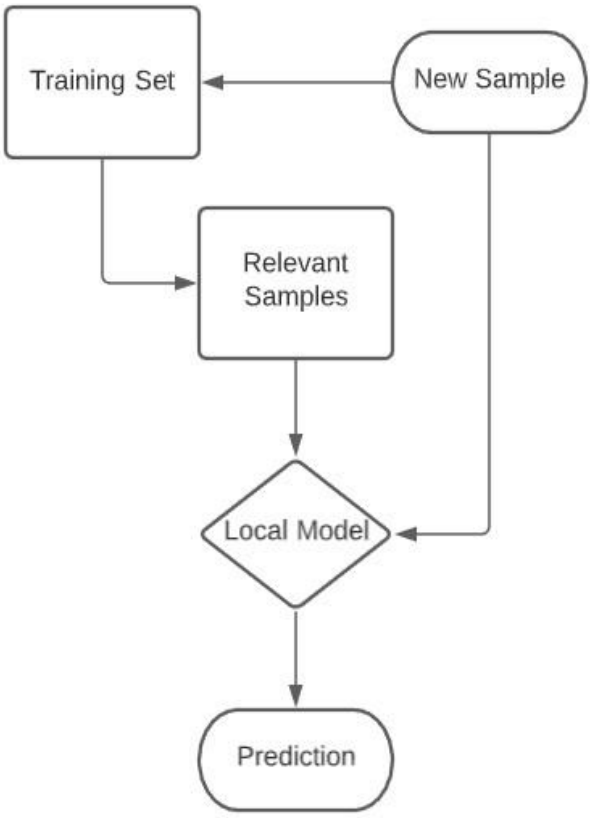


Figure 3.1. Schematic representation of JITL algorithm.

Determining the relevant samples is the key concept of the JITL. Relevance can be defined as having any kind of similarity with the query point [33]. Despite the number of publications, selecting relevant data is still a major challenge in JITL. The following subsection is a compilation of different sample selection methods.

3.1.1. Relevant Sample Selection in JITL

A local model is built to fit a data only in a small region around a query point. This model is specific to that query point. Hence, to improve the prediction performance, most

relevant samples should be selected. Relevance is meant to be capable of defining the relation between input and output data in that region [35]. Selection of relevant samples requires a similarity metric. Distance-based metrics are widely used for simplicity. Euclidean distance (ED) is the most commonly used and well-known distance metric.

If two samples that consist of p variables are $x = [x_1,\ x_2, \ldots, x_p]^T$ and $z = [z_1,\ z_2, \ldots, z_p]^T$, then a general representation of distance, known as Minkowski distance, is shown in Equation 3.1:

$$\Delta_m(x, z) = \left(\sum_{j=1}^{p} \left| x_j - z_j \right|^m \right)^{1/m} . \tag{3.1}$$

In equation 4.1, m determines the characteristic of the distance metric: m = 1 corresponds to Manhattan distance, while m = 2 corresponds to Euclidean distance. When $m \to \infty$, the corresponding distance metric is called Chebyshev distance, which is used to find the maximum distance between two samples in any coordinate dimension. Intuitively, as m increases, large values become more important because taking high powers of a large number exaggerates the values with respect to small numbers. There are advantages and disadvantages to each distance metric. For instance, Euclidean distance is quite sensitive to the outliers.

It is proposed that distance might be insufficient to explain the relevance, and angle based similarity criterion is used in addition to distance metric [37]. Equation 3.2 shows the angle based similarity.

$$cos(\theta) = \langle x, z \rangle / (\|x\|_2 \|z\|_2). \tag{3.2}$$

Both distance based similarity and angle based similarity are brought together in a single formula with a tuning parameter in Equation 3.3. Here, when $\omega = 0$, similarity only depends on angle between x and z; when $\omega = 1$, it is vice versa. The similarity formula mentioned in Equation 3.3 can be modified and combined with different techniques.

$$S_{xz} = \omega \exp\left(-\Delta_m(x,z)\right) + (1-\omega)cos(\theta). \quad (3.3)$$

In a more recent work, Mei et al. suggests using distance and angle based similarity in input space to select a group of similar samples. Then, distance based similarity in output space is used to eliminate irrelevant outputs among selected samples [37]. The suitable similarity criteria depends on the dataset. Euclidean distance might be the appropriate criteria for some datasets whereas angle based similarity might uncover the real similarity for other datasets. Yuan et al. proposed an ensemble JITL framework claiming that it is difficult to know the real similarity criteria for most processes, hence it might be useful to use different similarity criteria at the same time, consisting of Euclidian distance (Figure 3.2a), angle based distance (Figure 3.2b). Euclidean distance is simple and widely-used but correlation between variables is ignored when only ED is used. The correlation is taken into account when angle based similarity is used but irrelevant variables cannot be eliminated. Similarity in latent space deals with irrelevant variables, however, strong nonlinearities are not taken into account. The final prediction is made by taking weighted average of various predictions with respect to their training performance.

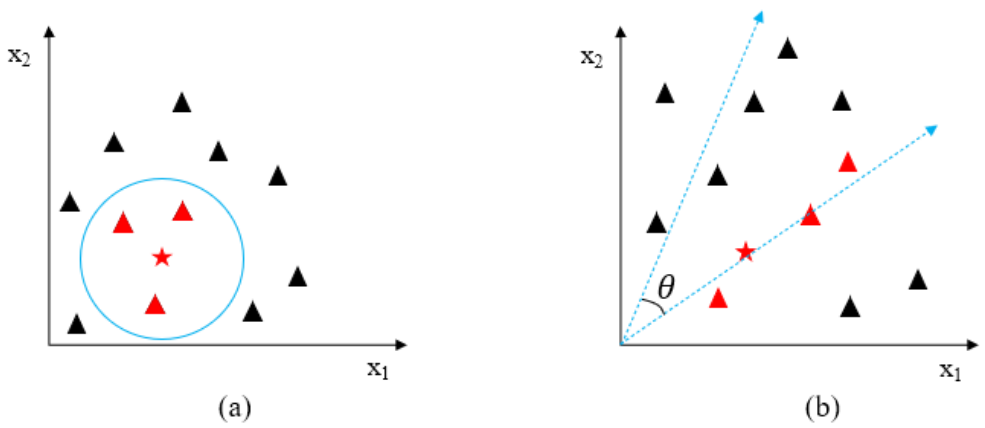


Figure 3.2. Sample selection with (a) Euclidean distance (b) angle-based similarity.

Fujiwara et al. considered correlations between samples and presented correlation based JITL modeling (CoJIT) [39]. In the proposed model, data set are divided into several sub data sets. These groups consist of successive samples, since successive points generally

are more correlated. Then, samples, which have a correlation structure most consistent with the query point, are selected. Figure 3.3 is an intuitive representation of the algorithm. Classical JITL modeling selects the nearest neighbors of the query point, whereas CoJIT selects the best sub data set according to their correlation.

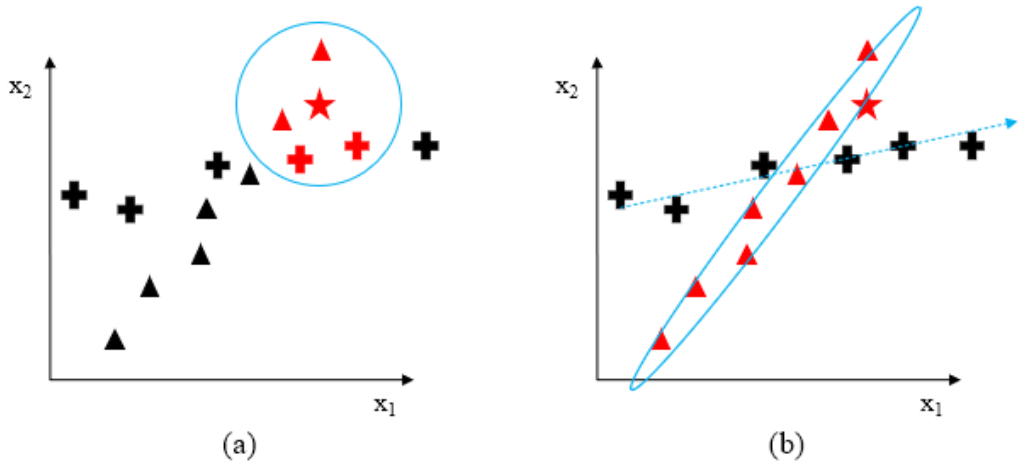


Figure 3.3. Schematic representation of (a) JITL and (b) CoJIT.

Zhang et al modified the traditional nearest neighbor rule by developing k surrounding neighbor [40]. After finding the closest sample in the training set to the query point, other samples are chosen in a way that leads to enclose the query point. For instance, the second closest point is not chosen, if it is in the same direction with the closest point in this method. Nock et al introduced symmetric nearest neighbor (sNN) rule [41]: Figure 3.4a and 3.4b shows the schematic representation of 1-NN and 1-sNN rule, respectively. The traditional 1-NN rule selects the closest point to x_1, whereas 1-sNN rule selects the closest point and the other two points that are in different directions but relatively further from the x_1. When the query point is x_2, 1-NN and 1-sNN select the same point because there are no other points in different directions.

3.2. Moving Window

Moving window (MW) is an adaptive method, in which a sliding set of points is selected to build a model. The selected set of points is aimed to have a maximum relevance for the current process operation [32]. The size of the sliding window is often a fixed number. When a new sample is acquired, it is added to the selected set of samples, and the oldest point is removed from this set. The most recent data point is assumed to be most relevant

with the current process characteristics. If the window size is taken to be small, MW is also computationally efficient method [43].

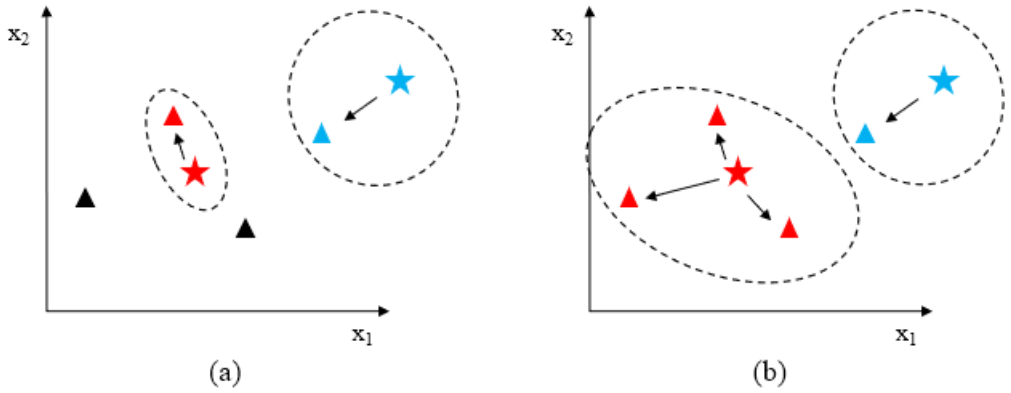


Figure 3.4. Schematic representation of (a) 1-NN and (b) 1-sNN.

One drawback is that moving window might not adapt to sudden changes in the process, since data points before the rapid change are still used. Additionally, if models are built in a steady-state region for a long time, they might give good predictions over a small region. Hence, storing a previously used set of window data is an option to counteract rapid changes in the process. A hypothesis testing may be used to determine whether a query point belongs to current operating zone [43].

Size of the sliding window is also an important parameter to be tuned. Prediction accuracy depends on the size because using large sliding block might be too slow to adapt changes in the process characteristics, while a too small sliding window might yield predictions with high variance [44].

In traditional moving window method, window data is updated for each newly obtained data. However, irregularly sampled quality variables and variable delays in measurements are also common things in industrial processes. Delayed update is proposed against continuous update in such cases [45]. Figure 3.5 is a simple illustration of update mechanism. In Figure 3.5a, newly obtained samples, which corresponds to 4 in temporal axis, is added to window data and, sample 1 is removed while in Figure 3.5b, sample 4 and 5 are added to window data together in delayed update moving window.

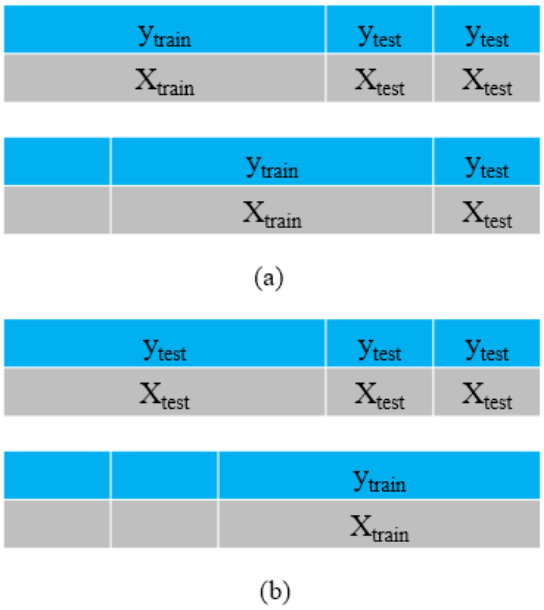


Figure 3.5. Schematic representation of (a) continuous update and (b) delayed update.

3.3. Time Difference

Time difference is an adaptive method that is built on the time difference of input data and output data. In time difference method, $\Delta \boldsymbol{x}(t)$ and $\Delta y(t)$ are the time differences that are determined from the present time and time points before the target time, $\Delta \boldsymbol{x}(t - \Delta t)$ and $\Delta y(t - \Delta t)$ [46,47]. Equation 3.4 and 3.5 shows how $\Delta \boldsymbol{x}(t)$ and $\Delta y(t)$ are calculated.

$$\Delta \boldsymbol{x}(t) = \boldsymbol{x}(t) - \boldsymbol{x}(t - \Delta t). \tag{3.4}$$

$$\Delta y(t) = y(t) - y(t - \Delta t). \tag{3.5}$$

Then, $\Delta \boldsymbol{x}(t)$ and $\Delta y(t)$ are model with desired statistical learning method. The time difference of the new data, $\Delta \boldsymbol{x}(t + \Delta t)$, is used to predict the time difference of $y(t + \Delta t)$, which is $\Delta y(t + \Delta t)$. Finally, $y(t + \Delta t)$ is determined through $\Delta y(t + \Delta t)$ as in Equation 3.6 and 3.7.

$$\Delta \boldsymbol{x}(t + \Delta t) = \boldsymbol{x}(t + \Delta t) - \boldsymbol{x}(t). \tag{3.6}$$

$$\hat{y}(t + \Delta t) = \Delta\hat{y}(t + \Delta t) + y(t). \quad (3.7)$$

Time difference models are proposed to overcome deterioration in prediction performance due to concept drifts and nonlinearity of process variables. Interval of time Δt should be carefully chosen because if the interval is too small, the time difference model might not catch the difference between separate process states [5].

4. DATA REDUCTION METHODS

Advances in computer technology and data storage devices have led to the capability of storing enormous amount of data. In real-life situations, there is noise, missing data, and outliers in raw data, which must be processed before using it. Hence, this data is not directly applicable to machine learning algorithms, and an expanding database may deteriorate the performance of a soft sensor [50]. Algorithms like JITL are directly affected by the size of a training set in computational time and predictive performance. When the training set is very large, building as a JITL model takes too much time [51]; in this case, focusing only on the informative part of the database can increase the efficiency of a soft sensor.

Data reduction (also referred to as instance selection or data selection in literature) is a technique to select, and store the relevant data, and introduce it to a statistical learning method. In other words, it is necessary to check how the predictive accuracy of a soft sensor is changed upon including an instance. As a general performance criterion, among the training set which render identical predictive performances, the one consisting of a minimum number of instances should be preferred. This may also be considered as an optimization problem, in which it is aimed to minimize data size, while prediction quality is aimed to maintained as a constraint [50].

Although data reduction methods offer significant improvements for machine learning algorithms, it has been mostly applied to classification problems [52-54]. There are a few examples that consider regression problems [5]. Hence, application of data reduction to regression task using local learning models remains an active research field. One method used in data reduction in regression problems, especially in instance-based learning, is to accept/reject a new instance based on checking whether a specific statistic of a novel instance is larger/smaller than a predefined threshold. This method is called threshold-based method throughout the current study. Criterion that shows the predictive ability (prediction error, RMSE, MAE), and nearest neighbor based proximity of instances are the traditional choices to be used in data reduction. Another alternative is to discretize response variable in order to apply data reduction methods for classification problem. In order to that, numerical response

variable of the training set is discretized, and a classification-based data reduction method is applied. If the instance is accepted, then restore the numerical value in selected instances [5]. In the following sections, we first discuss the main data reduction methods used for regression in the literature, then propose a number of new data reduction methods.

4.1. Data Reduction Methods for Regression in the Literature

4.1.1. Database monitoring index

Kaneko and Funatsu presented a database monitoring index as (DMI) for data reduction to increase the adaptive ability of a soft sensor [55]. This work is one of the few examples of data reduction for regression in literature, and used as the benchmark method in the current study. Their approach is to classify an instance using a function comprising the proximity of the novel instance to other instances in the response variable and predictor spaces. If a new instance spans a novel region in the response variable-predictor space, that have not been spanned by the instances in the training set before, this sample is added to the training set; otherwise, it is rejected. DMI is defined as the similarity of a new instance and another instance in the training set:

$$DMI = \frac{\left|y_i - y_j\right|^a}{sim(\boldsymbol{x_i}, \boldsymbol{x_j})}. \tag{4.1}$$

When response variable of the instance is measured, DMI values are determined between all point in the training set and the new instance. Here, similarity in predictor space, $sim(x_i, x_j)$, is measured by taking the inverse of the Euclidean distance (Equation 4.2), or Gaussian kernel (Equation 4.3). Hence, when two points get closer in the input space, their similarity increases.

$$sim(x_i, x_j) = 1/\sqrt{\left\|x_i - x_j\right\|^2}, \tag{4.2}$$

$$sim(x_i, x_j) = GK(x_i, x_j) = exp\left(-\gamma\left\|x_i - x_j\right\|^2\right). \tag{4.3}$$

Two parameters are to be determined in computing *DMI*: a and γ. The relative weight of the absolute difference between the values of the response variable to the similarity in X space is measured by a, while the Kernel width in Equation 4.3 is denoted by γ. Algorithm of DMI is shown in Figure 4.1. Here, the new instance can be deleted, or it is added to database coupled with the exclusion of the instance with minimum P_{DMI}. If the minimum DMI, denoted by $min(DMI)$, value of a novel instance to all instances already in the database exceeds the predefined threshold (P_{DMI}), the new instance is added to database. There is also an option to put an upper limit for the size of the database in this work. If the size of the database exceeds the limit, then the oldest point is deleted from the database. Here, the motivation is to keep the recently sampled points in the database, while removing the old points. This algorithm can be used for both JITL and MW methods. If MW is used, the size and the content of the window can be adjusted by DMI. For JITL, training set is monitored, and local models can be built as usual.

Require: New instance (x_j, y_j), and parameters $a, \gamma, P_{\mathrm{DMI}}$.

Data: Training set $T = [(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)]$

for $i = 1$ to N **do**

1- $sim(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2)$

2- $DMI = |y_i - y_j|^a / sim(x_i, x_j)$

if $min(DMI) > P_{\mathrm{DMI}}$ **then**

$T \leftarrow (x_j, y_j)$

end if

end for

Return T

Figure 4.1. *DMI* algorithm.

4.1.2. Edited Nearest Neighbor

Edited Nearest Neighbor is a well-known approach used for data reduction in classification problems. Gonzales et al. extended ENN to regression by using an adaptive threshold parameter, taken to be proportional to a constant parameter α, and the standard deviation of the response variable values in the nearest neighbors (see below) [5]. In this

algorithm, each point in the training set is examined separately. It starts taking a point (x_i, y_i), and removing it temporarily: Denoting the remaining training set by $T\backslash x_i$, and the nearest neighbors of x_i by S, i.e. $S = kNN(x_i, T\backslash x_i)$, and the removed point is predicted using $T\backslash x_i$, i.e. $\hat{y}(x_i) = f(T\backslash x_i, x_i)$ (Figure 4.2). If the prediction error is greater than the adaptive threshold value θ, then (x_i, y_i) is permanently removed from the training set. It is important to note that threshold value is not taken to be constant value, but depends on the spread of the values of the response variable in the local region. As the spread is smaller, absolute value of the prediction error is expected to be smaller; hence heteroscedasticity of the prediction errors is assumed. This process is repeated for each point in the training set. This algorithm does not focus on the acceptance of a new instance, but is mainly used as an outlier detector, or noise filter [5].

Require: α to determine threshold

Data: Training set $T = [(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)]$

for $i = 1$ to N **do**

1- $\hat{y}(x_i) = f(S, x_i)$

2- $S = kNN(x_i, T\backslash x_i)$

3- $\mathbf{y_S} = \{y(x_j), x_j \in S\}$

4- $\theta = \alpha \times std(\mathbf{y_S})$

if $|\hat{y}(x_i) - y(x_i)| > \theta$ **then**

$T \leftarrow T\backslash(x_1, y_1)$

end if

end for

Return T

Figure 4.2. Edited Nearest Neighbor.

4.1.3. Condensed Nearest Neighbor

Condensed Nearest Neighbor (CNN) differs from ENN in adding the new instance to database. If the prediction error is greater than the threshold, then the instance is included in the database. Otherwise, if the new instance is similar to instances in the database, then this

point is not accepted to the database. It can be said that this similarity-based logic is analogous to the one in the DMI method. Higher threshold value θ leads to acceptance of more instances in ENN whereas θ results in rejection of more instances in CNN [5]. The algorithm is shown in Figure 4.3.

Require: α to determine threshold

Data: Training set $T = [(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)]$

$P = \varnothing$

$P \leftarrow P \cup (x_1, y_1)$

for $i = 2$ to N **do**

1- $\hat{y}(x_i) = f(P, x_i)$

2- $S = kNN(x_i, T)$

3- $\mathbf{ys} = \{y(x_j), x_j \in S\}$

4- $\theta = \alpha \times std(\mathbf{y_S})$

if $|\hat{y}(x_i) - y(x_i)| > \theta$ **then**

$P \leftarrow P \cup (x_i, y_i)$

$T \leftarrow T \backslash (x_1, y_1)$

end if

end for

$T \leftarrow P$

Return T

Figure 4.3. Condensed Nearest Neighbor.

4.2. The Proposed Data Reduction Algorithms

The algorithms discussed above have some common points, such i) detection of similarity between input variables, ii) detection of proximity of response variables (the ones that can be referred as distance-based methods), and iii) evaluation using an error-related assessment criteria like threshold (this can be called error-threshold methods). In the current study, we aim to suggest a number of efficient algorithms combining these points, and check

the validity of these methods. The data reduction algorithm in DMI method is compared the ones that we suggest.

4.3. Data Reduction Using Average Distance (AD)

The first proposed method is data reduction using average distance (AD) algorithm, which is used to monitor and regulate the density of instances in the predictor space. The aim of algorithm is to attain an approximately homogenously distributed input space. A predefined threshold decides which point is accepted or rejected.

Require: New instance (x_j, y_j) and threshold θ

Data: Training set $T = [(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)]$

1- Find distances between the query point and the training set:

$d \leftarrow [d_1, d_2, \ldots, d_N]$

2- Find k nearest neighbors with the smallest $d_{(i)}, i = 1, 2, \ldots k$, in which the paranthesis in subscript refers to sorted observations in increasing order.

$\mathrm{NN} \leftarrow [(\mathrm{x}_{(1)}, \mathrm{y}_{(1)}), \ldots, (\mathrm{x}_{(k)}, \mathrm{y}_{(k)})]$

3- $d_{AD} = \sum_{m=1}^{k} d_m / \sqrt{k}$

4- $T \leftarrow (x_j, y_j)$

if $d_{AD} < \theta$ **then**

$T \leftarrow T \backslash (\mathrm{x}_{(1)}, \mathrm{y}_{(1)})$

end if

Return T

Figure 4.4. Average Distance Algorithm.

In AD, a metric named d_{AD}, which describes the data density in the neighborhood of the query point, is employed. To compute d_{AD}, the arithmetic mean of the NN distances are divided to square root of the number of the nearest neighbors $\sqrt{k}$. In this way, d_{AD} is "normalized" with respect to the size of the training set.

The detailed algorithm of AD method is shown in Figure 4.4. A new instance is obtained, and its nearest neighbors are found according to distance metric. These instances are gathered in NN. The closest point to the query point is CP. The set d consists of the distances between the query point and the nearest neighbors. At step 3, sum of these distances is divided to the number of nearest neighbors k, yielding d_{AD}. If the value of d_{AD} exceeds the threshold, then the new instance is included in the training set. This means that local region spanned by the nearest neighbors is scarce of observations. If d_{AD} is smaller than threshold, then, the query point is added to training set again, while removing the closest point to the query point from the database, i.e. density of the local region is maintained while keeping the database up to date.

4.4. Average Distance and Prediction Error

A possible drawback of AD method is that AD does not take the prediction accuracy of the query point into account. Average Distance and Prediction Error (ADPE) method considers both the density of the predictor space and the prediction performance of the nearest neighbors. The rationale behind ADPE is that if the existing samples in the region around a new instance is already predicted accurately, and density of samples is also large in that region, then including the new instance would not be required. However, if prediction errors of the nearest neighbors is large, and/or the density of samples is small, then the novel instance should be included in the training set. Hence, contribution of both prediction accuracy and density in the predictor space are taken into consideration in ADPE. There is also a parameter, α, which adjust the weights of these contributions.

Data: Training set $T = [(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)]$

for $i = 1$ to N **do**

$P \leftarrow T \setminus (x_i, y_i)$

1- Find distances between (x_i, y_i) and the training set:

$d \leftarrow [d_1, d_2, \ldots, d_{N-1}]$

2- Find k nearest neighbors:

$\mathrm{NN} \leftarrow [(\mathrm{x_1}, \mathrm{y_1}), \ldots, (\mathrm{x_k}, \mathrm{y_k})]$

$\hat{y}(x_i) = f(NN, x_i)$

Figure 4.5. Average Distance and Prediction Error-a Algorithm.

$PE_i = |\hat{y}(x_i) - y(x_i)|$

end for

Require: New instance (x_j, y_j), threshold θ, and parameter α

1- Find distances between the new instance and the training set:

$d \leftarrow [d_1, d_2, \ldots, d_N]$

2- Find k nearest neighbors, and extract the correspoding prediction errors:

$\mathrm{NN} \leftarrow [(\mathrm{x}_{(1)}, \mathrm{y}_{(1)}), \ldots, (\mathrm{x}_{(k)}, \mathrm{y}_{(k)})]$

$\mathrm{PE}_{NN} = [PE_{(1)}, PE_{(2)}, \ldots, PE_{(k)}]$

3- $d_{AD} = \sum_{m=1}^{k} d_m / \sqrt{k}$

4- $\overline{\mathrm{PE}} = avg(\mathrm{PE}_{NN})$

5- $d_{PE} = d_{AD} + \alpha \times \overline{\mathrm{PE}}$

6- $\hat{y}(x_i) = f(NN, x_j)$

7- $PE \leftarrow |\hat{y}(x_j) - y(x_j)|$

8- $T \leftarrow (x_j, y_j)$

if $d_{PE} < \theta$ **then**

$T \leftarrow T \backslash (\mathrm{x}_{(1)}, \mathrm{y}_{(1)})$

$PE \leftarrow PE \backslash PE_{(1)}$

end if

Return T

Figure 4.5. Average Distance and Prediction Error-a Algorithm (con't).

Figure 4.5 shows the algorithm of ADPE-a. Firstly, PE values of each point in the training set are determined. To do that, a sample is selected and nearest neighbors are searched in the training set, and a model is built with these samples. The chosen sample is predicted with this model, and prediction error ($\mathrm{PE_i}$) is determined. This value is written in prediction error vector. This process is iterated for each sample in the training set. For a query point, Nearest neighbors are determined as in AD algorithm. In addition, PE values of each of the nearest neighbor are concatenated in PE_{NN} vector. Then, d_{AD} is determined by dividing the summation of distances to the square root of the number of nearest neighbors. The average prediction error of the NN samples ($\overline{\mathrm{PE}}$) is determined by simply taking arithmetic mean of the PE_{NN} vector. Then, $\overline{\mathrm{PE}}$ is multiplied by α, and added to d_{AD} to form

the final metric d_{PE}. If α is taken to be too small, then the algorithm converges to AD algorithm. If α is taken to be large, then the contribution of the sample density in the neighborhood becomes insignificant. Hence, it is a very prominent parameter. If d_{PE} is greater than the threshold, the query point is accepted to the training set. If not, the query point is added to the training set again, but the closest point is removed from the database. PE value of the closest point is also removed from PE matrix.

The alternative to removing closest point is to sort the nearest neighbors according to the weighted sum of their distances and prediction errors, and to remove the sample with the smallest sum. This algorithm is called ADPE-b, and shown in Figure 4.6. ADPE-a and ADPE-b are identical to (and including) step 8, and only differentiates, when d_{PE} value is smaller than threshold: ADPE-a removes the closest point to the query point, while ADPE-b removes the samples with the minimum distance and prediction error.

Data: Training set $T = [(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)]$

for $i = 1$ to N **do**

$P \leftarrow T \setminus (x_i, y_i)$

1- Find distances between (x_i, y_i) and the training set:

$d \leftarrow [d_1, d_2, \ldots, d_{N-1}]$

2- Find k nearest neighbors:

$\mathrm{NN} \leftarrow [(\mathrm{x}_1, \mathrm{y}_1), \ldots, (\mathrm{x}_\mathrm{k}, \mathrm{y}_\mathrm{k})]$

$\hat{y}(x_i) = f(NN, x_i)$

$PE_i = |\hat{y}(x_i) - y(x_i)|$

end for

Require: New instance (x_j, y_j), threshold θ, and parameter α

1- Find distances between the new instance and the training set:

$d \leftarrow [d_1, d_2, \ldots, d_N]$

2- Find k nearest neighbors, and extract the correspoding prediction errors:

$\mathrm{NN} \leftarrow [(\mathrm{x}_{(1)}, \mathrm{y}_{(1)}), \ldots, (\mathrm{x}_{(k)}, \mathrm{y}_{(k)})]$

$\mathrm{PE}_{NN} = [PE_{(1)}, PE_{(2)}, \ldots, PE_{(k)}]$

Figure 4.6. Average Distance and Prediction Error-b Algorithm.

3- $d_{AD} = \sum_{m=1}^{k} d_m/\sqrt{k}$

4- $\overline{\mathrm{PE}} = avg(\mathrm{PE}_{NN})$

5- $d_{PE} = d_{AD} + \alpha \times \overline{\mathrm{PE}}$

6- $\hat{y}(x_i) = f(NN, x_j)$

7- $PE \leftarrow |\hat{y}(x_j) - y(x_j)|$

8- $T \leftarrow (x_j, y_j)$

if $d_{PE} < \theta$ **then**

$d_{PE-NN} = [\, d_{(1)}/\sqrt{k} + PE_{(1)} \quad d_{(2)}/\sqrt{k} + PE_{(2)} \ldots \; d_{(k)}/\sqrt{k} + PE_{(k)}]$

Find the m^{th} element such that $d_{PE-NN}(\mathrm{m}) = \min(d_{PE-NN})$.

$T \leftarrow T \backslash (\mathrm{x}_{(\mathrm{m})}, \mathrm{y}_{(\mathrm{m})})$

$PE \leftarrow PE \backslash PE_{(m)}$

end if

Return T

Figure 4.6. Average Distance and Prediction Error-b Algorithm (con't).

To sum up, there are 4 different methods that are examined. DMI method is the literature reference and AD, ADPE-a and ADPE-b are applied on 4 different data sets.

5. DATASETS USED IN THE STUDY

There are four different datasets that are used examine predictive performances of the proposed dataset reduction methods: an artificial (synthetic) dataset (ART), air quality data (BNZ), Sulfur Recovery Unit (SRU) dataset, and Waste Water Treatment Plant (WWTP) dataset. The first dataset is produced via computer simulation as in Kaneko et al [57], while the other ones are publicly available real industrial datasets. A summary of the datasets is given in Table 5.1.

Table 5.1. NN, γ, and σ^2 values for LSSVR for each data set.

Dataset	Number of Samples in Training Set	Number of Samples in Test Set	Number of Input Variables
ART1	500	500	2
ART2	1500	500	2
BNZ1	500	3000	8
BNZ2	3000	3000	8
SRU 1	5000	5075	35
SRU 2	5000	5075	35
WWTP	60	300	11

5.1. ART Dataset

Equation 5.1 shows the equation used to generate samples in ART dataset.

$$y = \sin(x_1)\cos(x_2) + 0.1x_1 + \in, \tag{5.1}$$

Normal random errors ($\in$) with zero mean and 0.01 variance are added to generated the final y values. Here, x_1 and x_2 are deemed to be process variables, while the noisy y measurements correspond to the quality variables. In order to see the effect of initial size of historical data on the predictive performance of the proposed data reduction methods, two different training sets are used for ART, named ART1 and ART2. Training sets of 500 and 1500 are used in ART1 and ART2, respectively, while using the same test sets.

5.2. BNZ Dataset

BNZ dataset contains hourly averaged air quality data which are obtained by air pollution monitoring station in Italy [56]. Output variable is benzene concentration whereas input variables are concentrations of CO, non metanic hydrocarbons, NO_x, NO_2, O_3, temperature, relative humidity, and absolute humidity, respectively. BNZ1 and BNZ2 are also created to examine the effect of training set size. BNZ1 consists of 500 training samples whereas there are 3000 training samples in the BNZ2. There are 3000 samples in tests sets for both.

5.3. SRU Dataset

SRU dataset has two response variables, and they are represented by SRUa (hydrogen sulfate) and SRUb (sulfur dioxide). Experimental data is collected at ERG PETROLI refinery [57]. There are 10075 samples in total. The training set consist of 5000 samples and there 5075 samples in the test set. There are 5 predictors: the gas flow in MEA zone and SWS zone, air flow in MEA 1, MEA 2, and SWS zone. Additionally, 30 more inputs are generated from the lagged values of 5 predictors. Lagged timesteps are 1 to 6.

5.4. WWTP Dataset

Lastly, WWTP data set is used to evaluate the performances of Lasso and LSSVR. There 360 samples in this dataset. The training set consists of 60 samples and there are 300 samples in the test set [58]. There also 11 predictors in this dataset.

6. RESULTS AND DISCUSSION

Four data reduction methods, DMI, AD, JITL, ADPE-a, and ADPE-b, are assessed using a linear learner Lasso, and a non-linear learner LSSVR on four different datasets. It is important to note that the number of nearest neighbors (NN), and the hyperparamaters of the learners and data reduction methods should be conveniently tuned using the training data for each dataset. The size of the nearest neighbors (NN), and the hyperparamaters of the learners are discussed in Section 6.1, and prediction performance of JITL using the optimum parameters is discussed in Section 6.2; it should be noted that data reduction methods are not used in these sections. In Sections 6.3 to 6.6, sensitivity of the parameters in the four data reduction methods are determined on different datasets. Finally, Section 6.7 compares the predictive accuracy of the learners using various data reduction methods.

6.1. Adjusting the NN for JITL and Learner Parameters

Multi-dimensional grid searches are conducted for each dataset to determine the optimum size of NN and learner parameters, i.e. λ in Lasso (Eqn. 2.27), and γ and σ^2 in LSSVR (Eqns. 2.37 and 2.41). As an example, a grid search over NN and λ for ART dataset is shown in Figure 6.1, in which the NN, λ couple corresponding to the minimum RMSE is deemed to be the optimum parameter set.

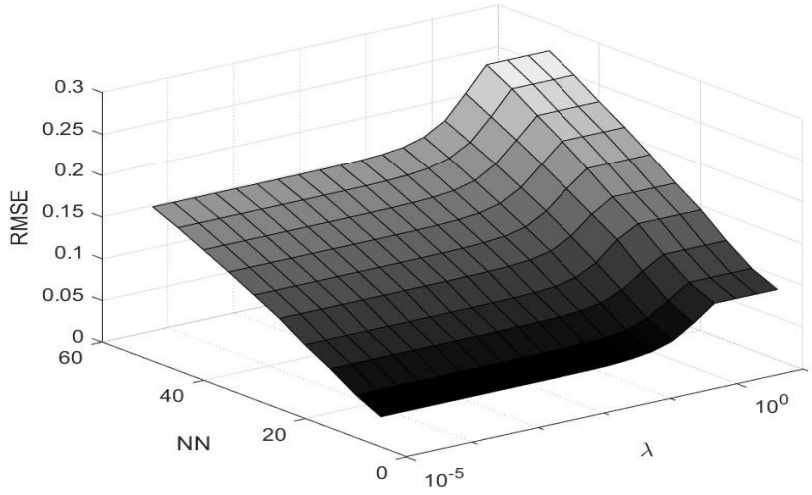


Figure 6.1. NN and λ parameter search for ART dataset.

Since ART dataset has two predictors, it is possible to visualize how learner parameters affect the fit surface. Figure 6.2 and 6.3 show the effects of σ^2 and γ on the surface created by the LSSVR model, respectively. Increasing σ^2 makes the fit smoother whereas at σ^2 = 2 it fits the data very well, while increasing hyperparameter γ leads to a better fit.

The optimum sets of {NN, λ} and {NN, γ, σ^2} values are given for Lasso and LSSVR learners for all datasets in Table 6.1 and 6.2, respectively. It should be further noted that LSSVM parameters are tuned by using LS-SVMlab Toolbox in MATLAB.

Table 6.1. NN and λ values for Lasso for each data set.

Dataset	**NN**	λ
ART	5	0.0127
BNZ	10	0.0150
SRU 1	10	0.0060
SRU 2	15	0.0030
WWTP	40	0.0500

Table 6.2. NN, γ, and σ^2 values for LSSVR for each data set.

Dataset	**NN**	γ	σ^2
ART	270	$8x10^3$	1.7^2
BNZ	350	10	1600
SRU 1	10	10^6	150
SRU 2	20	10^6	160
WWTP	15	10^{16}	2.7

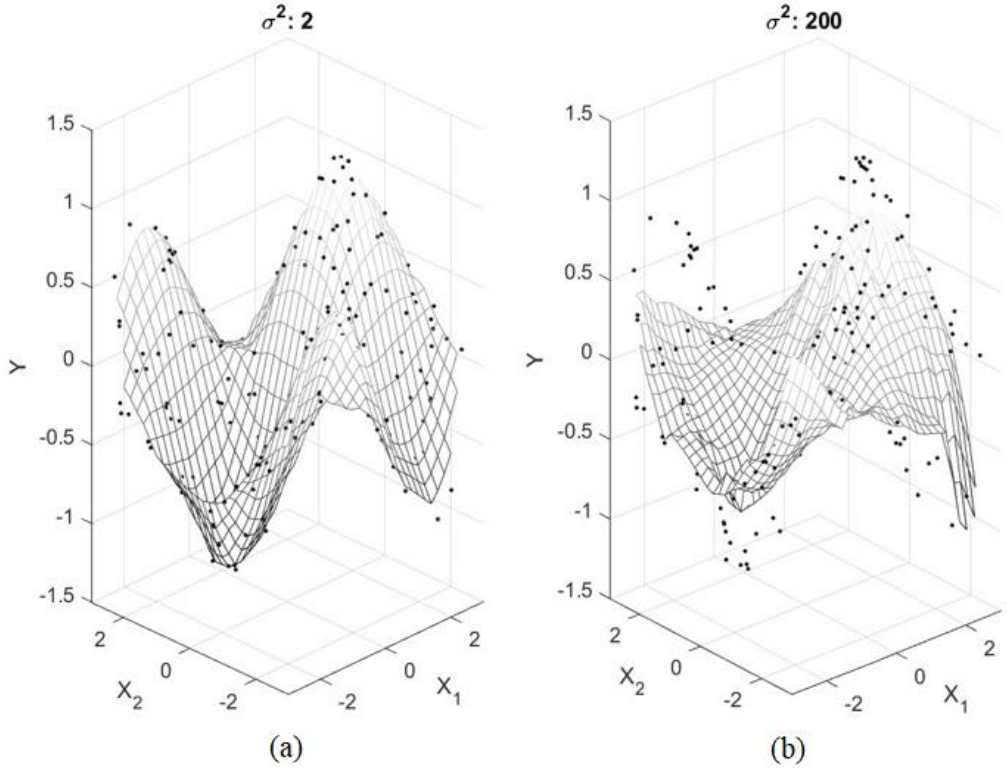


Figure 6.2. The effect of σ^2 on fitting for (a) σ^2=2 and (b) σ^2=200.

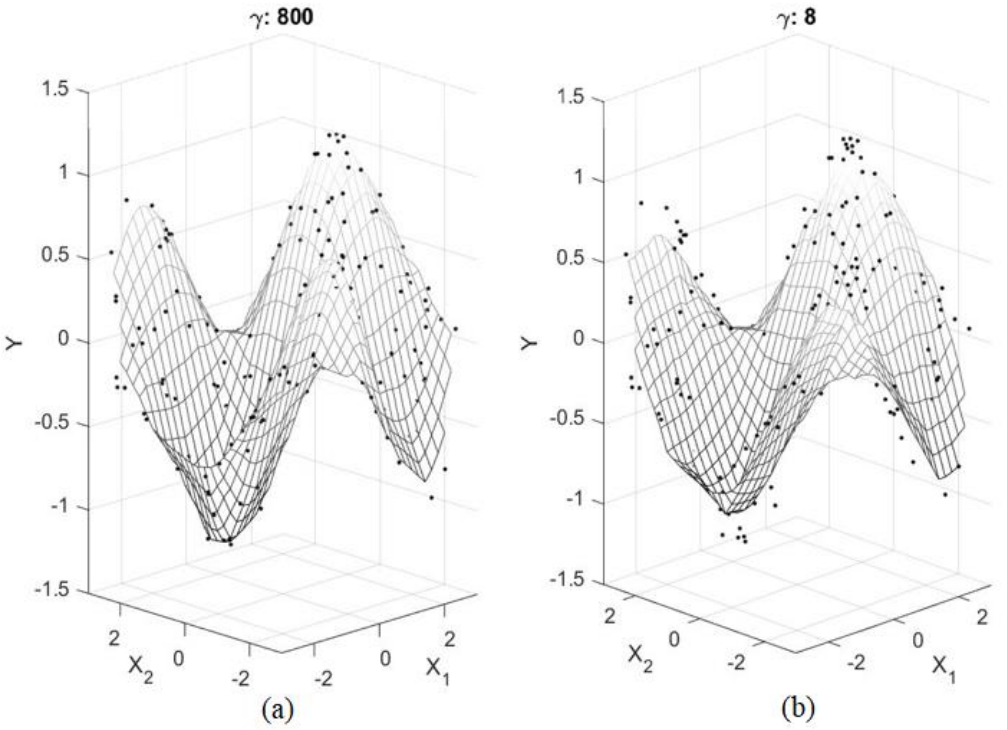


Figure 6.3. The effect of γ on fitting for (a) γ=800 and (b) γ =8.

6.2. Prediction Performance of JITL models using Lasso and LSSVR

The prediction performance of Lasso and LSSVR, using the optimum parameter values determined in the previous section, are compared in four different datasets, and the results are tabulated in Table 6.3. It is seen that LSSVR has smaller prediction error compared to Lasso in all datasets. Furthermore, Lasso seems to be sensitive to the initial database size; prediction accuracy difference between the two methods is more pronounced for ART1 and BNZ1 (smaller initial databases) compared to ART2 and BNZ2 (larger initial databases, see Table 5.1). For WWTP, SRUa and SRUb datasets, prediction accuracy of Lasso and LSSVR is rather close.

Table 6.3. RMSE values obtained via JITL using Lasso and LSSVR.

Dataset	Lasso	LSSVR
ART1	0.0218	0.0103
ART2	0.0147	0.0101
BNZ1	0.0836	0.0012
BNZ2	0.0669	0.0006
SRUa	0.0197	0.0176
SRUb	0.0228	0.0210
WWTP	0.0222	0.0209

6.3. Parameter Tuning in DMI

Out of the four data reduction methods to be assessed in the current study, DMI is to be used as a benchmark method, while AD, JITL, ADPE-a, and ADPE-b are novel proposed methods (see Section 4). DMI has two important parameters: $\boldsymbol{P}_{\mathbf{DMI}}$ and a (see Section 4.1.1). $\boldsymbol{P}_{\mathbf{DMI}}$ is the equivalent of the threshold parameter in other algorithms**,** while a determines the proportionality of the effect of similarity in input space to the effect of proximity between output variables. For small values of a, the similarity between response variables becomes

insignificant. As for similarity in input space increases, DMI values become smaller. Hence, the probability of to being under $\boldsymbol{P_{DMI}}$ increases, and the new sample is not included in the database. For large *a* value, the proximity between response variables becomes important.

6.3.1. DMI on ART1 Dataset

In Figure 6.4, both RMSE and % excluded points are plotted with respect to parameters $\boldsymbol{P_{DMI}}$ and *a* using Lasso for ART1 dataset. Here, % excluded points stands for how many points are rejected, or removed from the whole database. 0% of excluded points means that each new sample is accepted to the training set. The upper limit of the % excluded points is determined by the total size of the training set and test set. For instance, if the sizes of the training set and test set are equal and all points are rejected in the test set, then the maximum value of the % excluded points become 50%. If the size of test set is larger than training set, it is larger than 50%, if all points in the test set are rejected. Otherwise, % excluded points is smaller than 50% in the same situation. Figure 6.4 shows that RMSE is directly related to % excluded points except for one region. When P_{DMI} is large and *a* is between 0.1 and 1, DMI method shows somewhat competitive results although all points are rejected in this region.

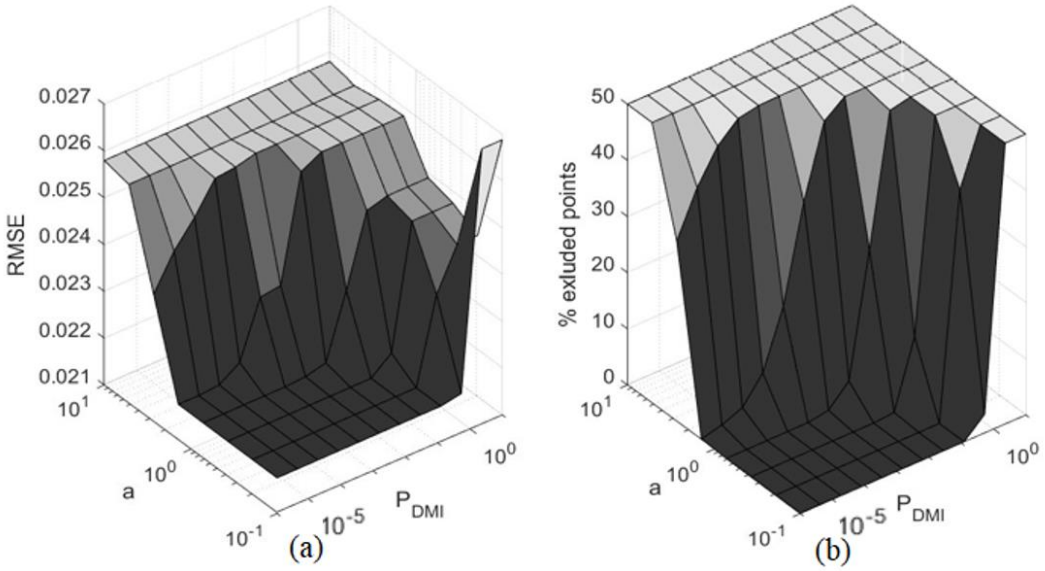


Figure 6.4. Grid search for parameters of DMI method using Lasso for ART1 for (a) RMSE and (b) % excluded points.

Figure 6.5 shows a more detailed view of the region mentioned above. Here, the dashed lines are the lower and the upper limits of the observed values in Figure 6.4. When P_{DMI} is 10, all points are rejected for each a value. However, RMSE fluctuates as a increases. As mentioned before, when a is between 0.1 and 1 and P_{DMI} is between 0.5 and 10, RMSE is roughly average of the best and worst results. According to Figure 6.4, the best results are observed, not surprisingly, when all points are accepted, and the worst results are seen when a is large. Hence, RMSE values around 0.024 can be seen as acceptable because all points in the test set are rejected, and database is not extended. Large a value indicates that proximity between response variables is more important than the closeness in the input space.

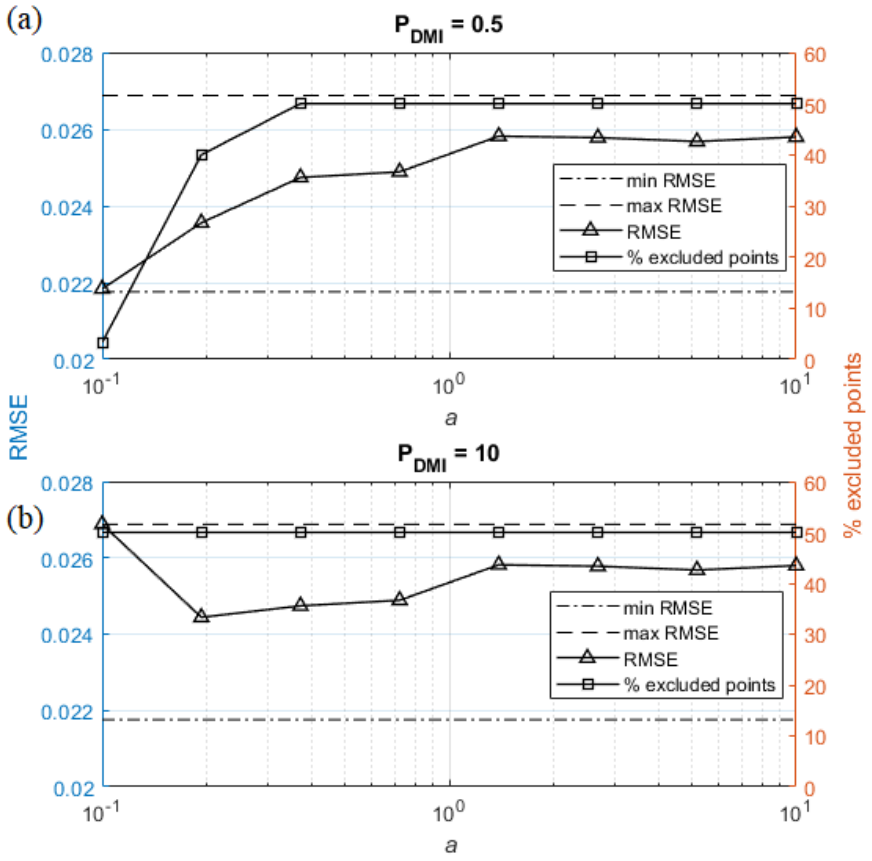


Figure 6.5. Performance of DMI method with respect to a using Lasso for ART1 for (a) P_{DMI}=0.5 and (b)P_{DMI}=10.

Figure 6.6 shows the result for DMI method when LSSVR is used for ART1. Although results show a change in RMSE values as % excluded points increases, the difference from Lasso results is insignificant. The increase in the error from the best result to the worst is

nearly 2.5%. If P_{DMI} is adjusted larger than 0.5 and a is between 0.1 and 1, all points are rejected and the increase in the error is smaller than 1%.

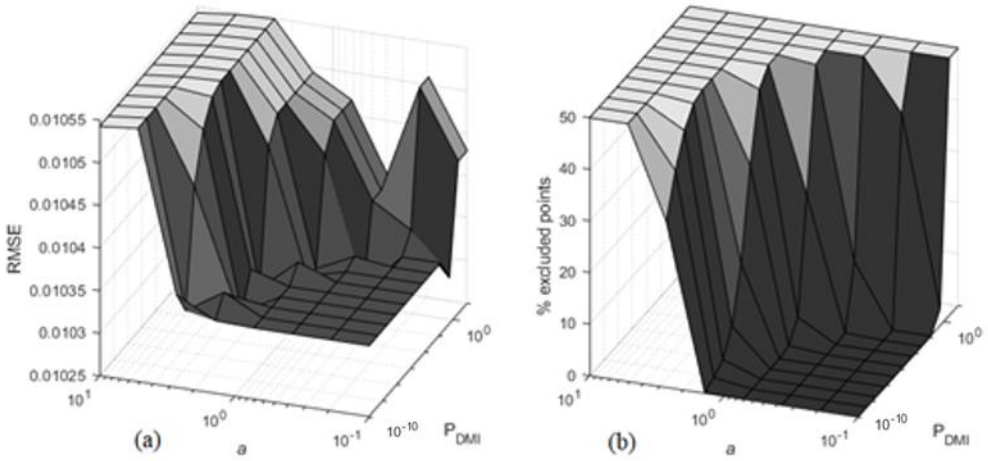


Figure 6.6. Performance of DMI method with respect to a using LSSVR for ART1 for (a) RMSE and (b) % excluded points.

6.3.2. DMI on BNZ1 Dataset

Figure 6.7 shows parameter search for BNZ1 dataset. The most distinctive region is where P_{DMI} is around 1 and a is small. The % excluded points surface shows the full rejection regions and its reflection to RMSE surface is quite different. Especially, when P_{DMI} is large, decreasing a leads to an increase in RMSE. For large a, different P_{DMI} values do not affect RMSE significantly when all instances are rejected in this region. Hence, when DMI method is used large a value can be chosen. Moderate a value that seem more suitable for LSSVR

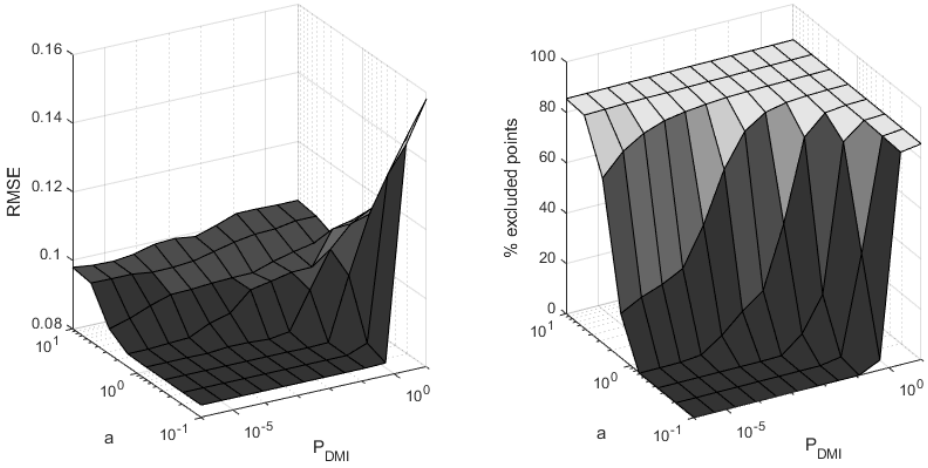


Figure 6.7. Grid search for parameters of DMI method using Lasso for BNZ1 for (a) RMSE and (b) % excluded points.

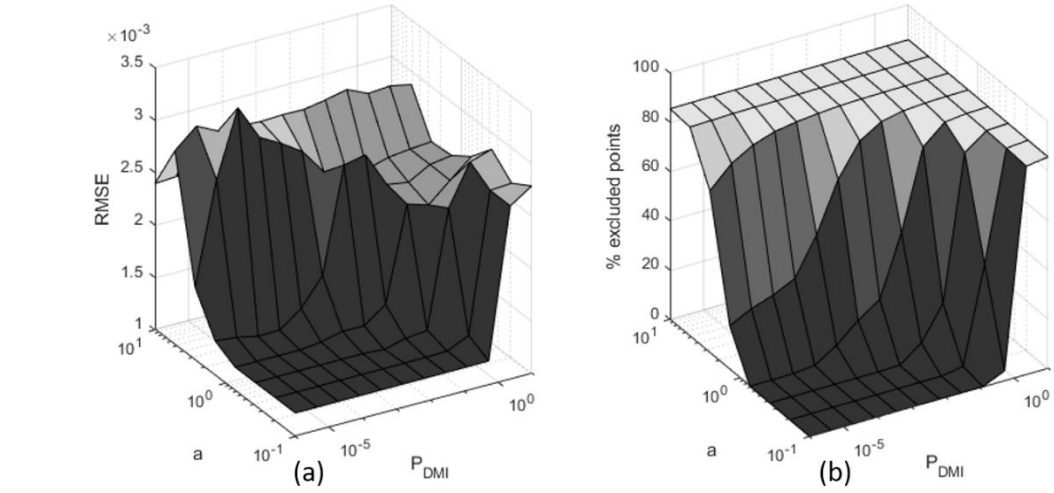


Figure 6.8. Grid search for parameters of DMI method using LSSVR for BNZ1 for (a) RMSE and (b) % excluded points.

6.3.3. DMI on SRU Dataset

Figure 6.9 shows the parameter search for DMI method for SRUa dataset. If the region where all instances are rejected is examined, the worst results are seen when both *a* and P_{DMI} is large. In addition, when P_{DMI} is fixed at 1, increasing *a* leads to a decreasing trend in RMSE although % excluded points is the same. LSSVR reacts different to DMI method for SRUa as Figure 6.10 shows. Grid search proves that large *a* shows competitive results despite all instances are rejected. Deterioration in prediction quality from JITL result is only 3% in this region. Hence. large *a* is a suitable parameter for DMI method.

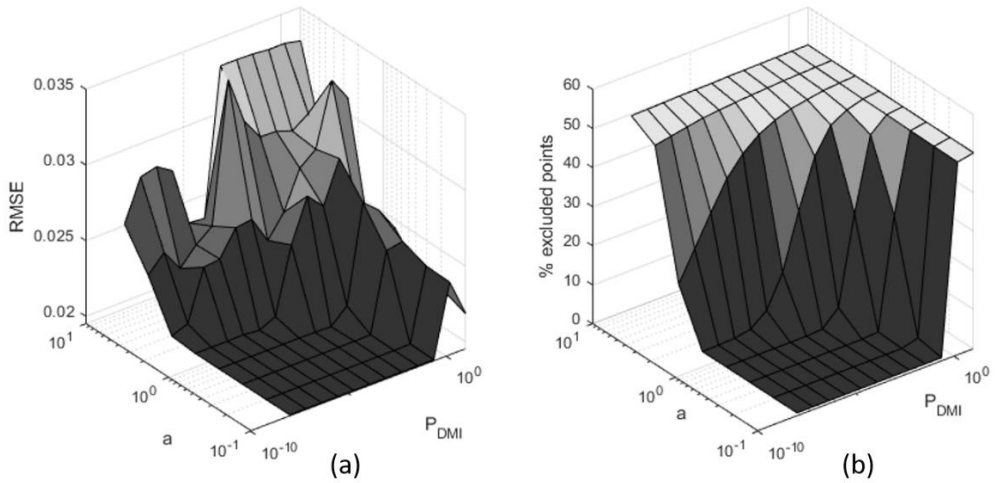


Figure 6.9. Grid search for parameters of DMI method using Lasso for SRUa for (a) RMSE and (b) % excluded points.

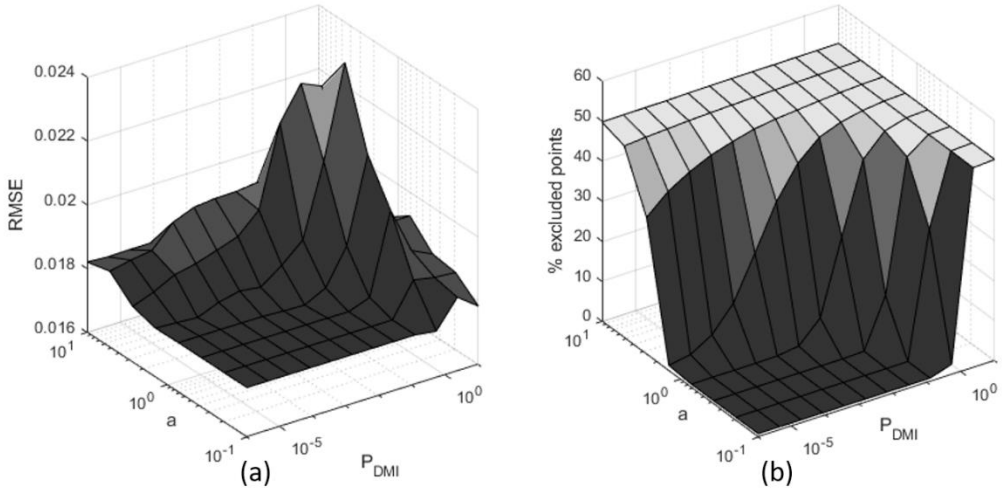


Figure 6.10. Grid search for parameters of DMI method using LSSVR for SRUa for (a) RMSE and (b) % excluded points.

When SRUb is examined for DMI method, Lasso results show that increasing *a* has a negative effect on prediction performance if P_{DMI} is fixed at a large number. Best result is achieved when P_{DMI} is large and *a* is roughly 0.1 (Figure 6.11). Figure 6.12 shows the search for DMI method using LSSVR for SRUb. Increasing *a* has a negative effect on prediction performance just like Lasso. RMSE is lower than RMSE of JITL when P_{DMI} is fixed at 10 and *a* is between 0.1 and 1.3 In addition, the prediction performance is also better than JITL when P_{DMI} is 2.3 and *a* is between 0.1 and 0.7.

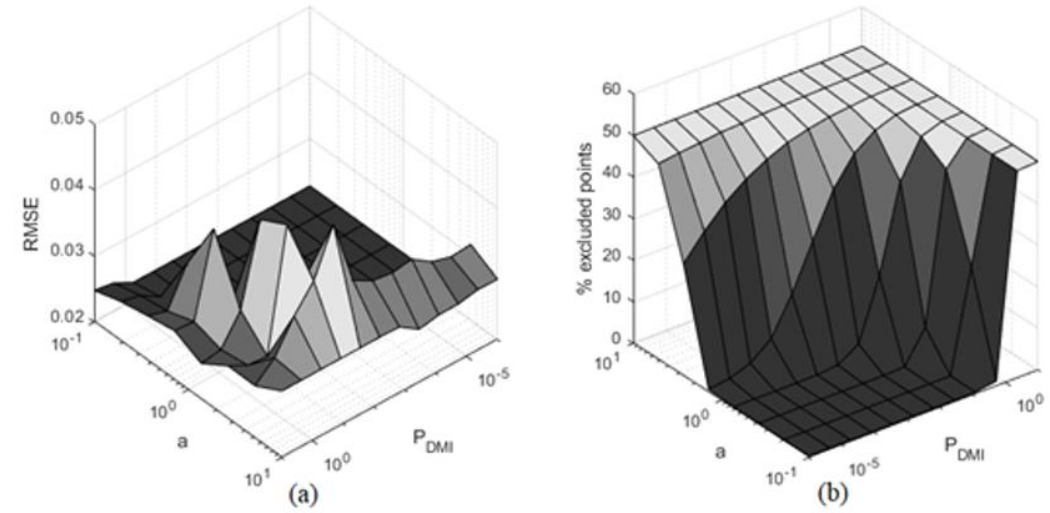


Figure 6.11. Grid search for parameters of DMI method using Lasso for SRUb for (a) RMSE and (b) % excluded points.

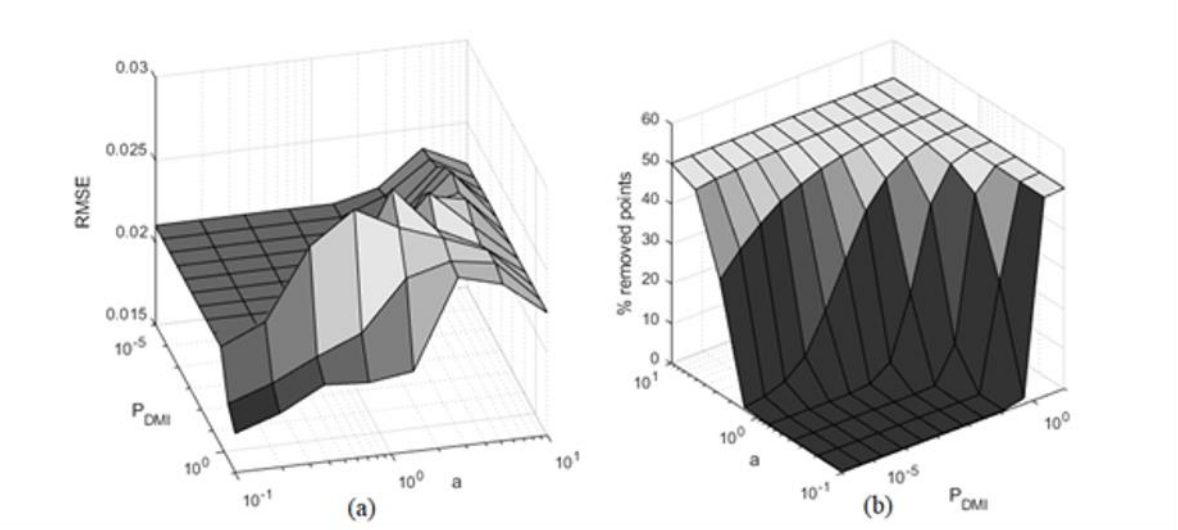


Figure 6.12. Grid search for parameters of DMI method using LSSVR for SRUb for (a) RMSE and (b) % excluded points.

6.3.4. DMI on WWTP Dataset

Figure 6.13 shows that the quality of predictions for WWTP data directly depends on the size of the training set when Lasso is used. However, when a is around 1 and P_{DMI} is low, RMSE is close to full acceptance even for % excluded points is between 20% and 50%. Hence, rejecting some points does not deteriorate prediction quality significantly in this region. Figure 6.14 shows that the prediction performance of LSSVR is slightly better than Lasso for WWTP dataset, and rejecting some points seriously deteriorates the prediction performance of LSSVR.

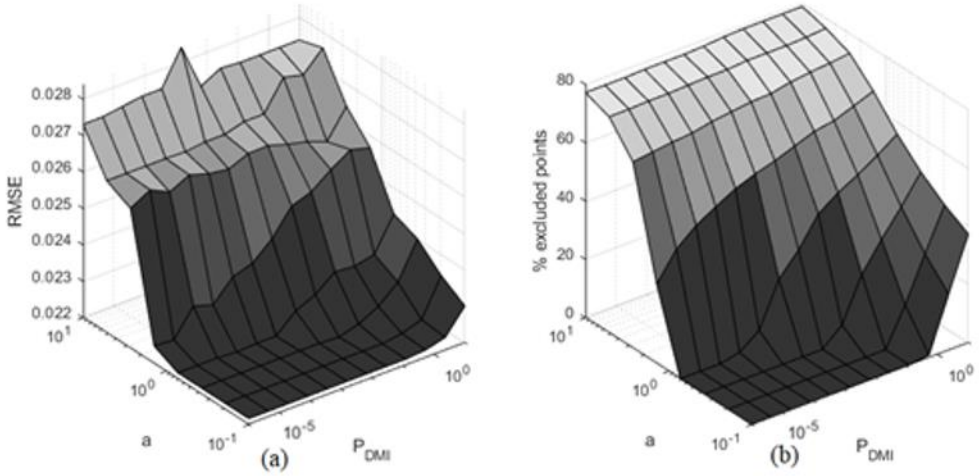


Figure 6.13. Grid search for parameters of DMI method using Lasso for WWTP for (a) RMSE and (b) % excluded points.

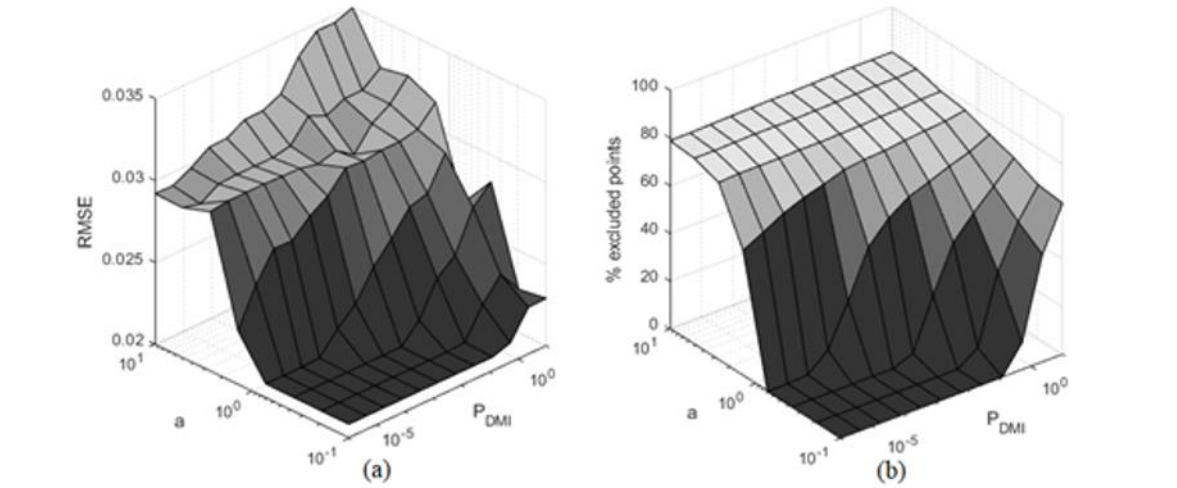


Figure 6.14. Grid search for parameters of DMI method using LSSVR for WWTP for (a) RMSE and (b) % excluded points.

6.4. Parameter Tuning in AD

Parameter search for AD method is simpler because it has a single threshold parameter θ (see Section 4.3).

6.4.1. AD on ART1 Dataset

Figures 6.15 and 6.16 show that the highest prediction accuracy is achieved for θ between 0.04 and 0.08, corresponding to 20% of the data reduction. Hence, it is interesting that not including some of the new samples to the database may increase prediction accuracy.

6.4.2. AD on BNZ1 Dataset

Figure 6.17 shows the effect of the threshold of AD on % excluded points and RMSE when Lasso is used. The predictive performance of JITL is maintained up to ~15% excluded points. In addition, predictions deteriorate only 10% even for 45% excluded points. Hence, AD can be accepted as successful until 45% excluded points. Then, RMSE increases linearly as θ increases. LSSVR follows a similar trend as seen in Figure 6.18. The learner maintains its predictive ability to some extent, and then RMSE significantly rises.

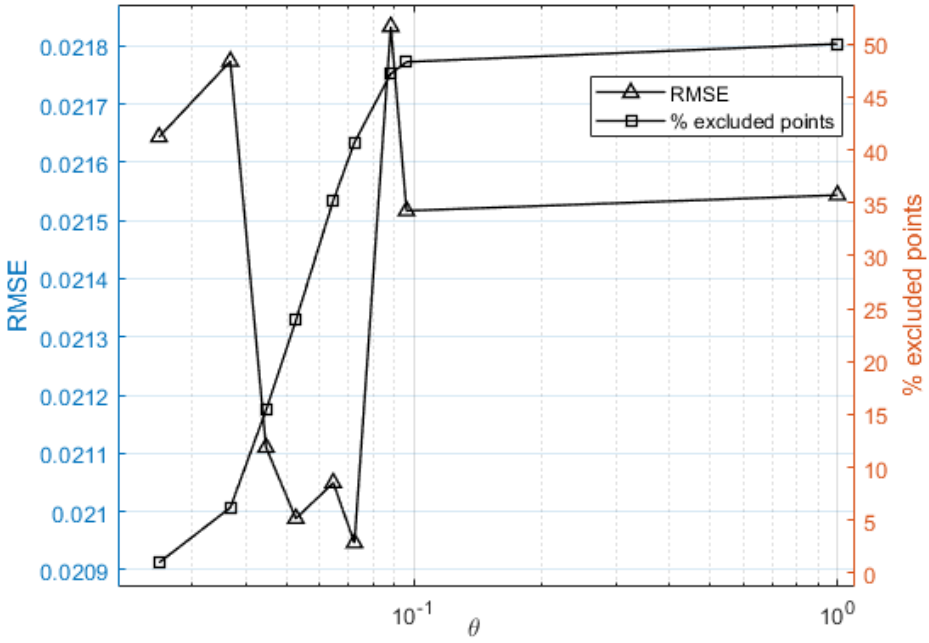


Figure 6.15. Parameter search for Average Distance with respect to θ using Lasso for ART1.

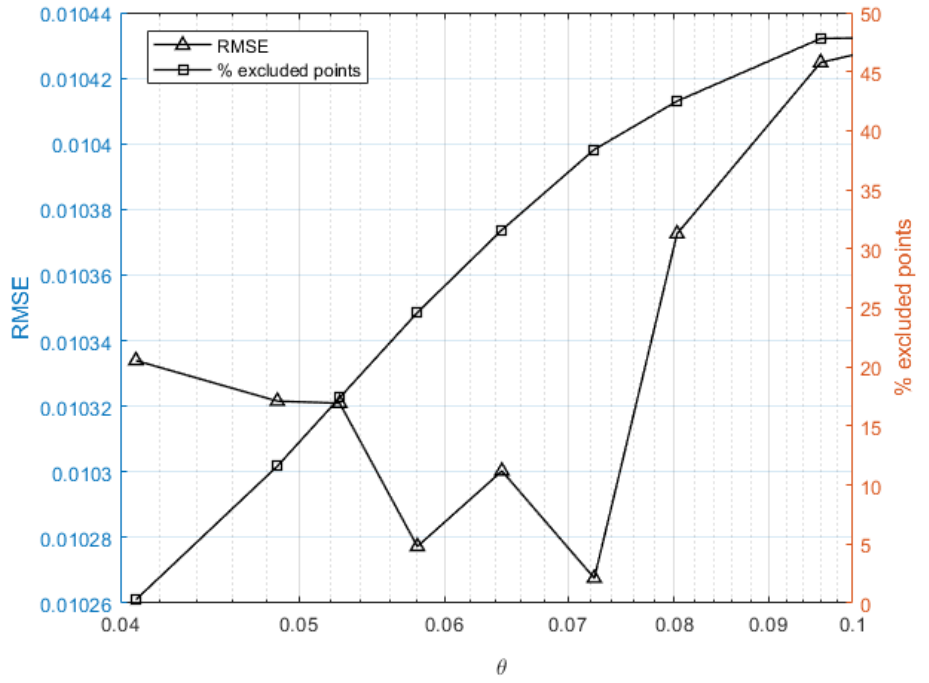


Figure 6.16. Parameter search for Average Distance with respect to θ using LSSVR for ART1.

It should be noted that same θ stands for different % excluded points, though training set and test set are the same for both cases. The reason is that size of the nearest neighbors are different for Lasso and LSSSVR. Hence, average of distances is different for each

algorithm. LSSVR also maintains its prediction accuracy up to 45% excluded points. Although prediction quality of both Lasso and LSSVR deteriorates significantly through 80% excluded points, the worst case for LSSVR is still much better than Lasso predictions.

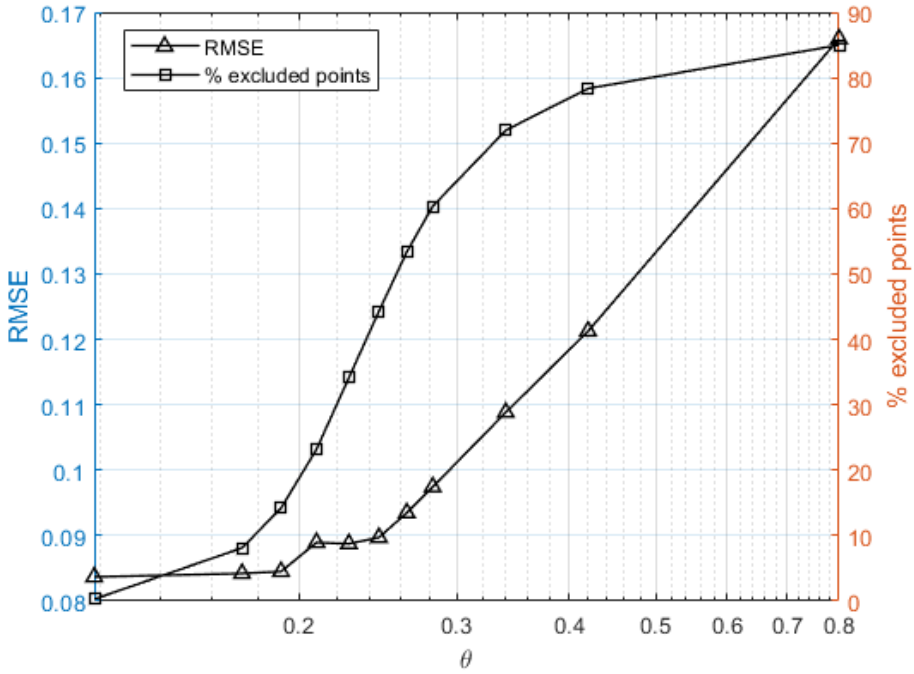


Figure 6.17. Parameter search for AD with respect to θ using Lasso for BNZ1.

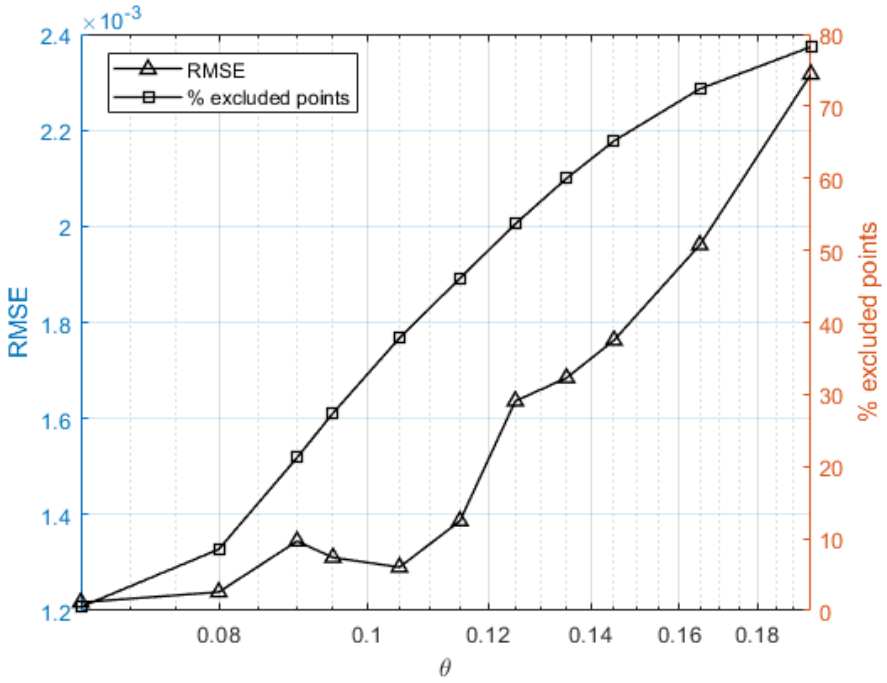


Figure 6.18. Parameter search for AD with respect to θ using LSSVR for BNZ1.

6.4.3. AD on SRU Dataset

Figure 6.19 shows the performance of Lasso when AD is used for SRUa. It is very important that AD can give better results while rejecting some instances: RMSE of 7% excluded points and 15% excluded points are lower than RMSE of JITL. In addition, predictive performance is still very similar to JITL at 25% excluded points. This means that rejection every one of four instances does not deteriorate prediction quality. Lastly, RMSE is slightly higher than that of JITL at 30% excluded points. AD is also quite effective when LSSVR is used for SRUa. It shows better RMSE than JITL at 22% excluded points (Figure 6.20). It also keeps its prediction quality with a small amount loss till 40% excluded points. Prediction accuracy deteriorates by ~3% at 35% excluded points, and there is 5% deterioration at 40% excluded points. Hence, it can be concluded that AD algorithm is quite effective when it is applied to SRUa dataset for both Lasso and LSSVR

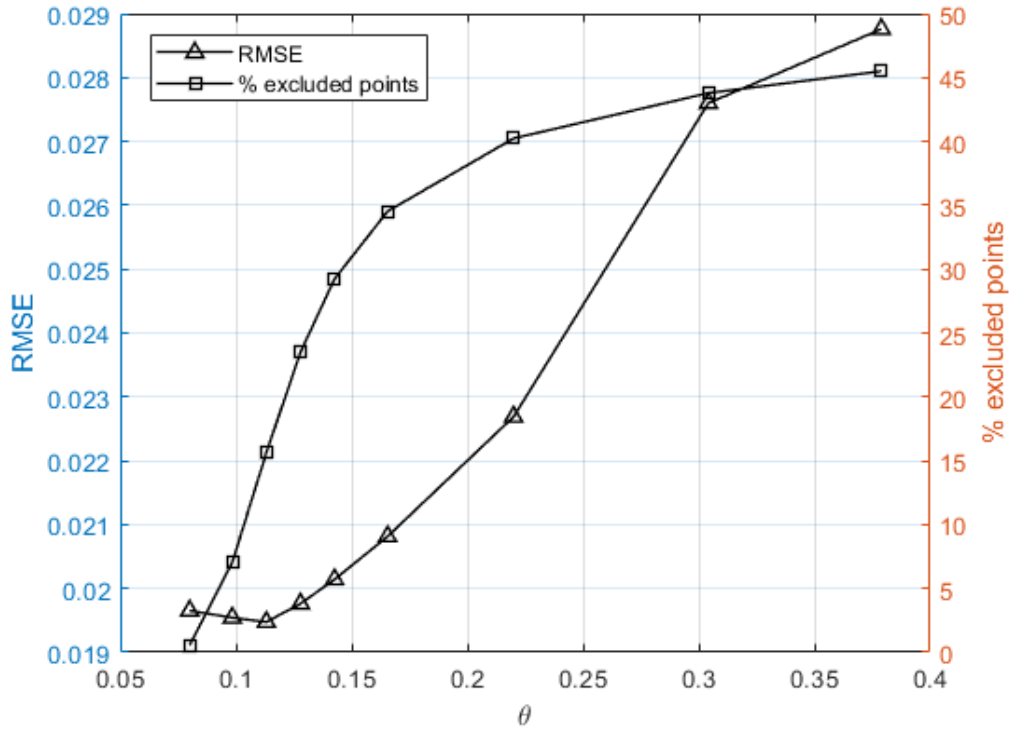


Figure 6.19. Parameter search for AD with respect to θ using Lasso for SRUa.

Figure 6.21 shows that AD can give lower RMSE than RMSE of JITL for 25% excluded points and it can keep its predictive performance till there for SRUb. Then, RMSE

increases with % excluded points. At 30% excluded points, it still gives results around 0.023. Hence, it is ideal to apply 30% data reduction if Lasso is used.

Figure 6.22 shows the effect of AD on RMSE and % excluded points for SRUb when LSSVR is used. RMSE significantly decreases when % excluded points is 50. RMSE rises along with % excluded points till 50%. However, when it reaches 50% excluded points, prediction quality is significantly better than JITL.

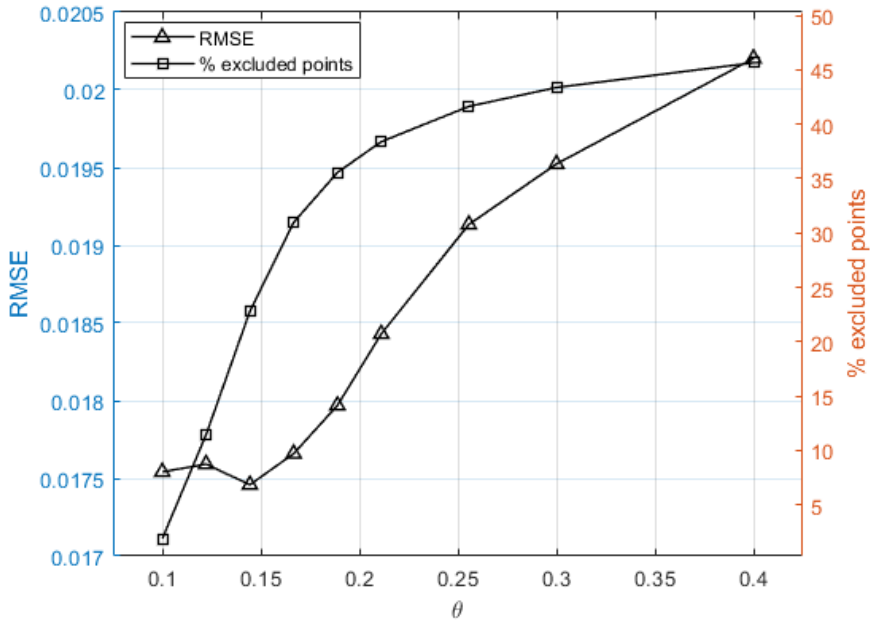


Figure 6.20. Parameter search for AD with respect to θ using LSSVR for SRUa.

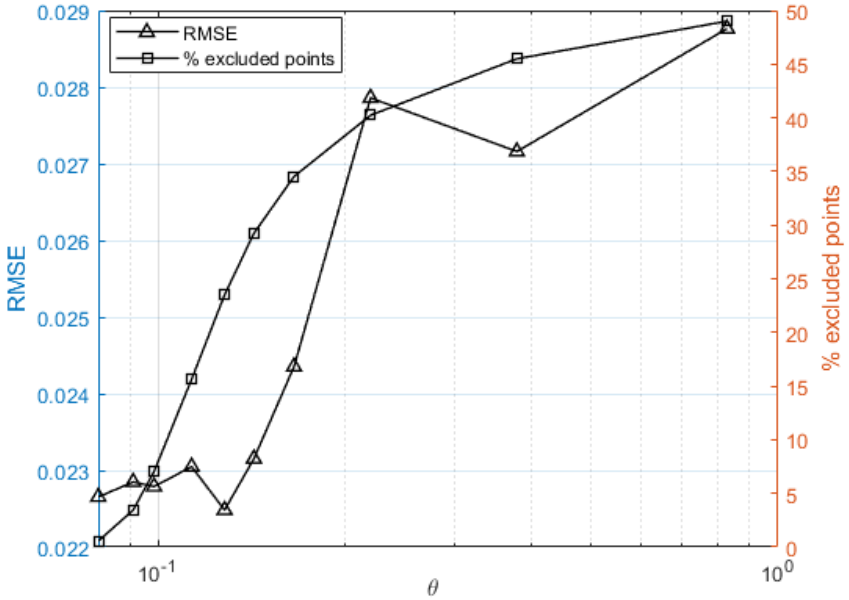


Figure 6.21. Parameter search for AD with respect to θ using Lasso for SRUb.

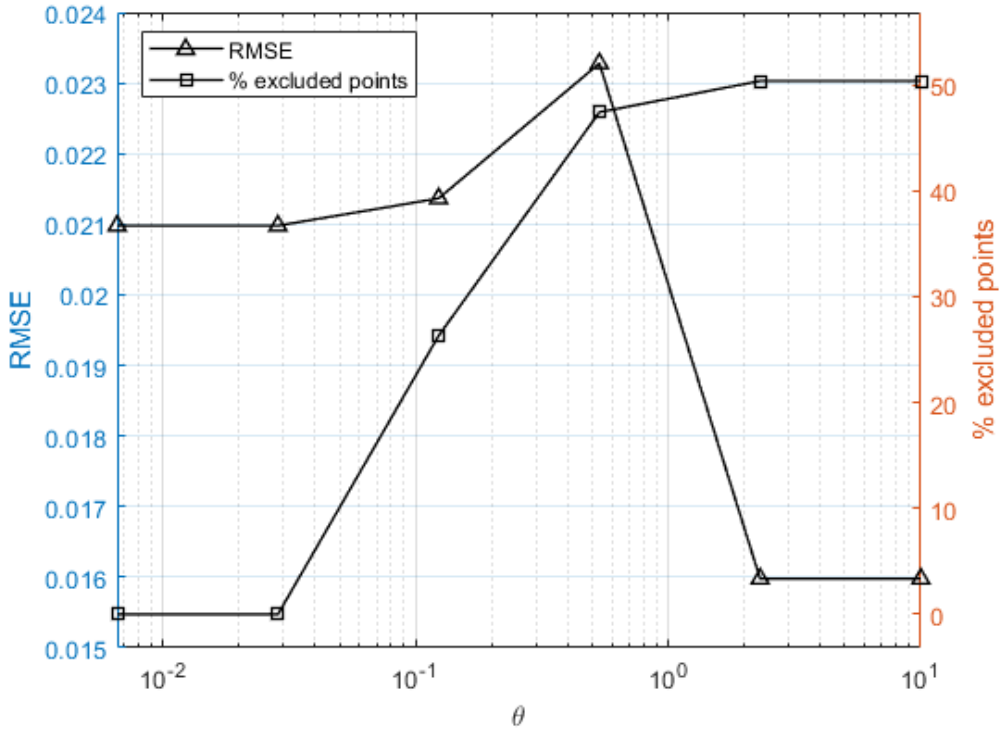


Figure 6.22. Parameter search for AD with respect to θ using LSSVR for SRUb.

6.4.4. AD on WWTP Dataset

AD algorithm works well up to 40% excluded points as can be seen in Figure 6.23. RMSE is also roughly 0.025 for 80% excluded points which can be accepted as a successful data reduction. If it is desired to maintain the predictive performance of JITL, AD algorithm can successfully eliminate nearly 25% of the test set without any loss in the predictive ability. In addition, at 25% excluded points, RMSE is lower than RMSE of JITL. This is an example of increasing predictive performance while keeping database smaller. Very large θ leads to both full rejection and a significant increase in RMSE. Figure 6.24 shows the parameter search for AD when LSSVR is used. LSSVR can also keep its predictive ability to 25% excluded points without any loss in prediction performance. In addition, RMSE is still quite small between 30% and 40% excluded points, and at 65% excluded points, RMSE is only 10% larger than the RMSE of JITL However, RMSE significantly rises after θ is 1. The performance of LSSVR is better than Lasso for small θ. However, when θ is large RMSE is large than 0.036 when all points are rejected. Hence, some of the specific samples seem to strongly affect the prediction quality of LSSVR.

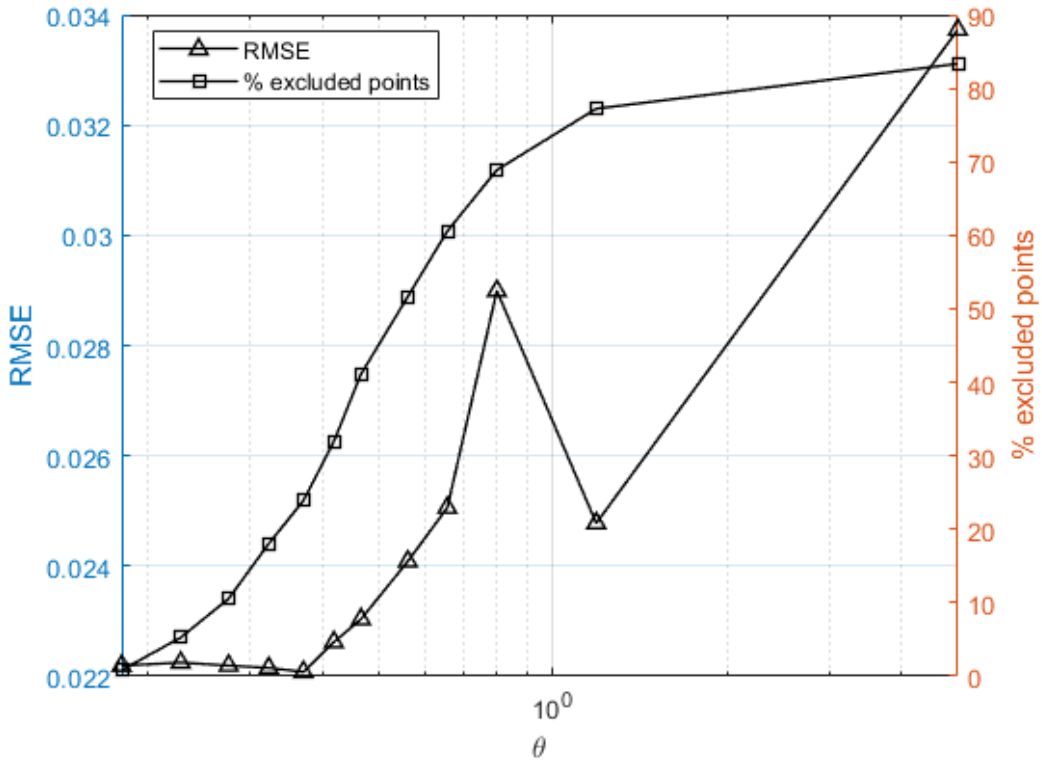


Figure 6.23. Parameter search for Average Distance with respect to θ using Lasso for WWTP.

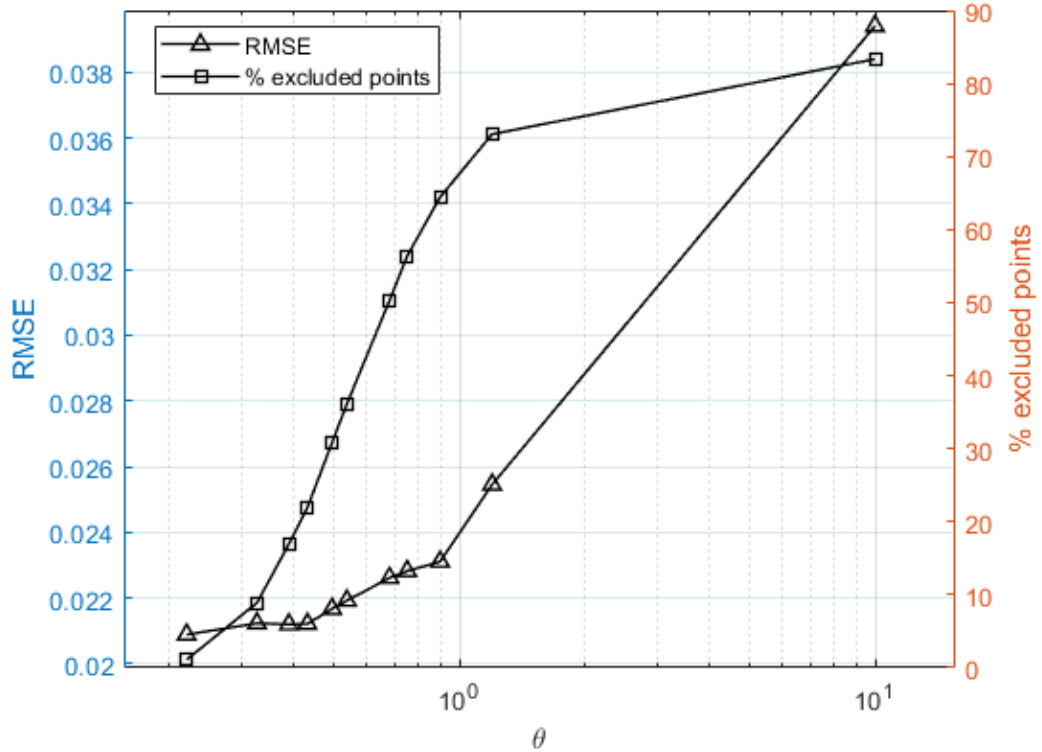


Figure 6.24. Parameter search for Average Distance with respect to θ using LSSVR for WWTP.

6.5. Parameter Tuning in ADPE-a

ADPE-a consists of two parameters (θ and α in Figure 4.5). Details of the application on BNZ1 and WWTP datasets are given in Appendix A1-A2, and A3-A4, respectively. Application on ART1 and SRU datasets are presented below.

6.5.1. ADPE-a on ART1 Dataset

Figures 6.25 and 6.26 for Lasso and LSSVR, respectively, show rather similar trends to those obtained for the data reduction methods discussed above. Increasing α increases the contribution of prediction error on rejection criteria. Hence, smaller α makes ADPE algorithms resemble AD more. The best results are achieved for roughly 30% rejection route that stands for different θ and α values. The lowest RMSE is seen when α is small and θ is 0.9. Figure 6.26 shows results for ADPE-a when LSSVR is used on ART1. The effect of parameters on RMSE is very limited. Nonetheless, excluding some of the dataset may still improve prediction results for small a values.

6.5.2. ADPE-a on SRU Dataset

Figure 6.27 shows the effect of α on both % excluded points and RMSE. When threshold is fixed at 0.23, increasing α naturally decreases % excluded points. RMSE follows a similar path with % excluded points. When θ = 1.5, although % excluded points is 50, RMSE is below 0.02. Hence, α between 1-5 can be chosen optimal parameters for SRUa. When ADPE-a is applied, LSSVR shows significantly better results than JITL when all instances are rejected (Figure 6.28). It is observed when α is between 1 and 10 when θ is fixed at 1.5. This is the most effective example so far about usefulness of data reduction methods. Hence, small α can be chosen to apply ADPE-a.

The effect of α on ADPE-a using Lasso is examined in Figure 6.29 for SRUb. Increasing α leads to a decrease in % excluded points and, this reflects to RMSE. In Addition, RMSE is lower than RMSE of JITL for 12% excluded points. Hence, α can be chosen around 50 to apply ADPE-a with Lasso. Figure 6.30 show the search for ADPE-a when LSSVR is used for SRUb. LSSVR can also give lower RMSE than RMSE of JITL if α is around 100.

However, % excluded points is only 3% at this point. α greater than 40 can be optimum parameter for ADPE-a.

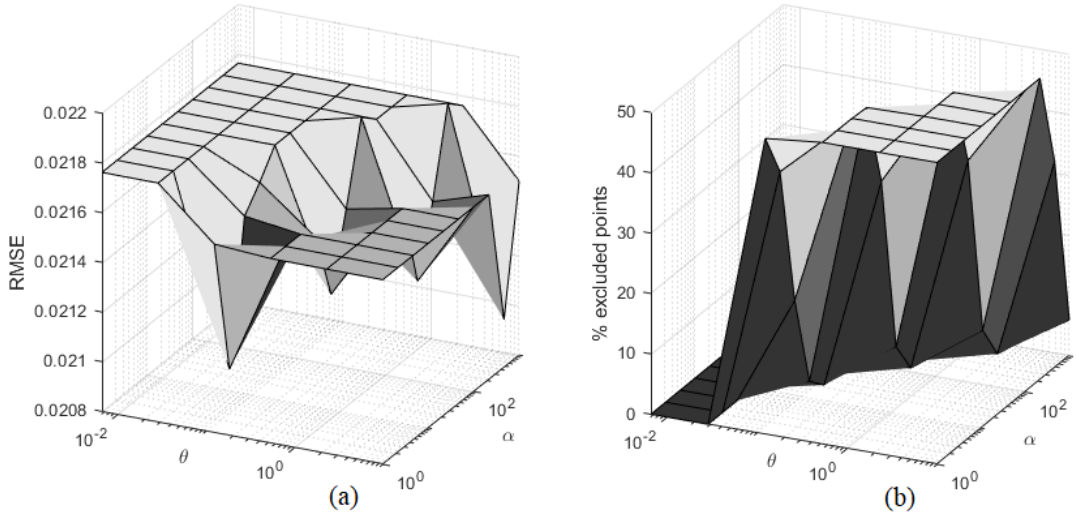


Figure 6.25. Parameter search for ADPE-a with respect to θ and α using Lasso for ART1 for (a) RMSE and (b) % excluded points.

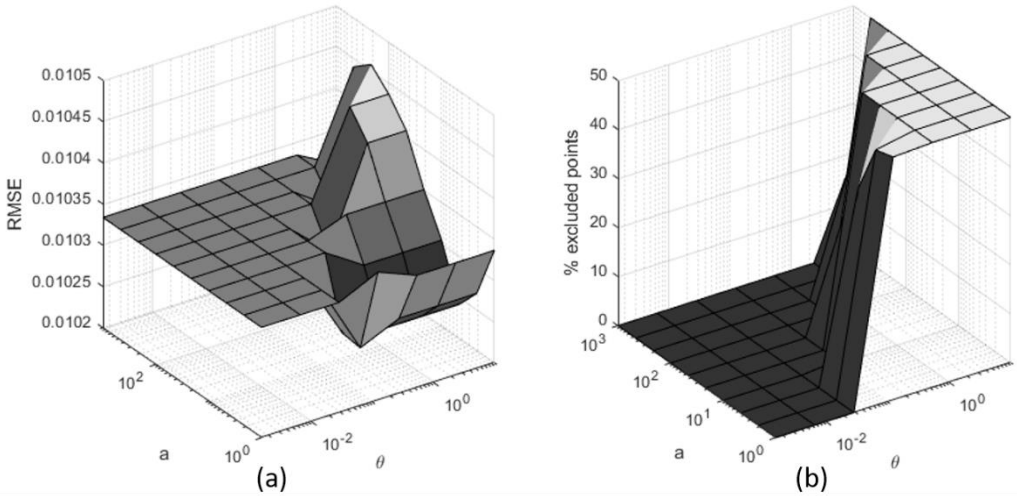


Figure 6.26. Parameter search for ADPE-a with respect to θ and α using LSSVR for ART1 for (a) RMSE and (b) % excluded points.

6.1. Parameter Tuning in ADPE-b

ADPE-v consists of two parameters (θ and α in Figure 4.6). Details of the application on on ART1, SRUa and SRUb datasets are given in Appendix A5-A6, A7-A8, A9-A10, respectively. Application on BNZ1 and WWTP datasets are presented below.

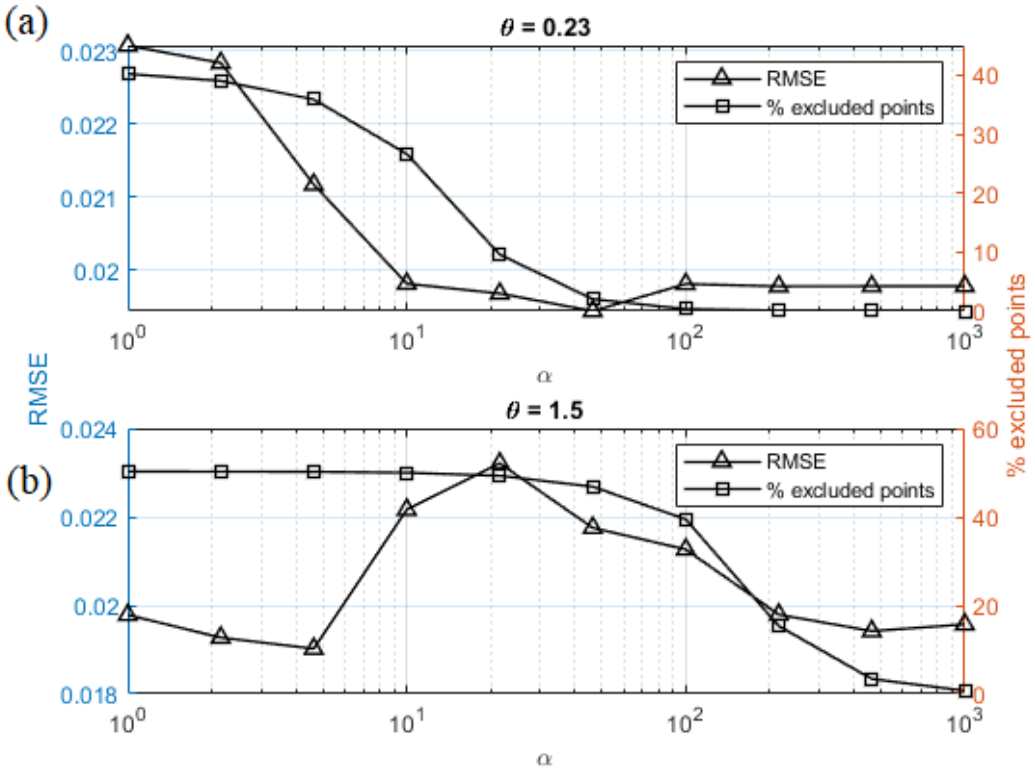


Figure 6.27. Parameter search for ADPE-a with respect to α using Lasso for SRUa for (a) $\theta = 0.23$ and (b) θ=1.5.

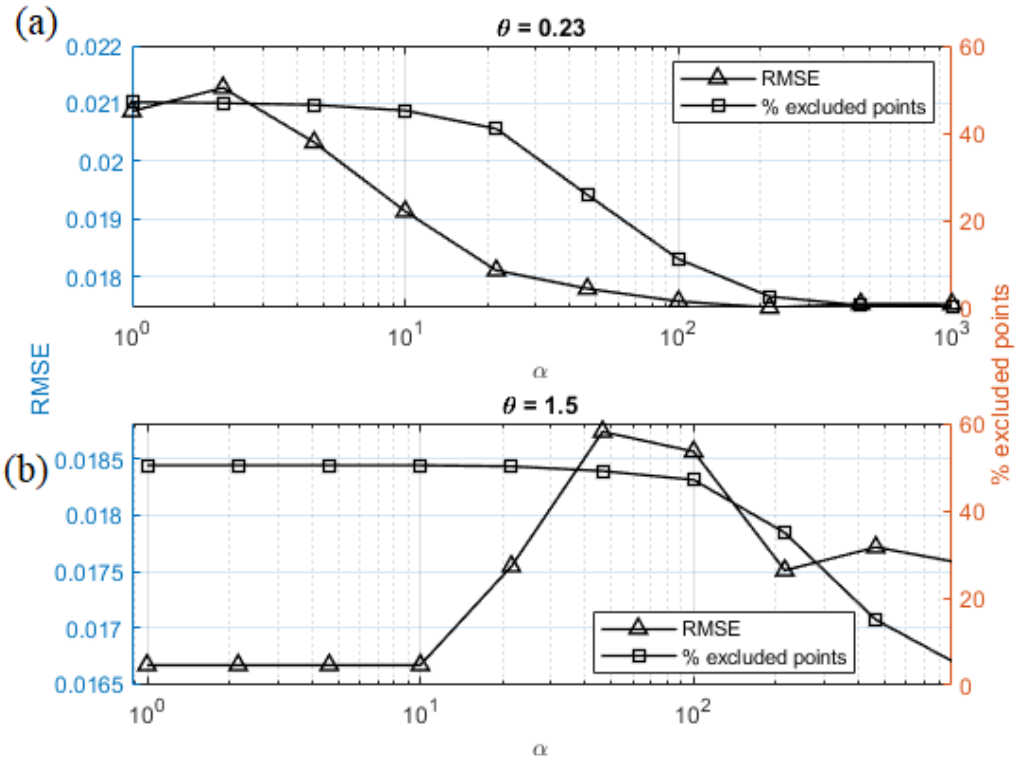


Figure 6.28. Parameter search for ADPE-a with respect to α using LSSVR for SRUa for (a) $\theta = 0.23$ and (b) θ=1.5.

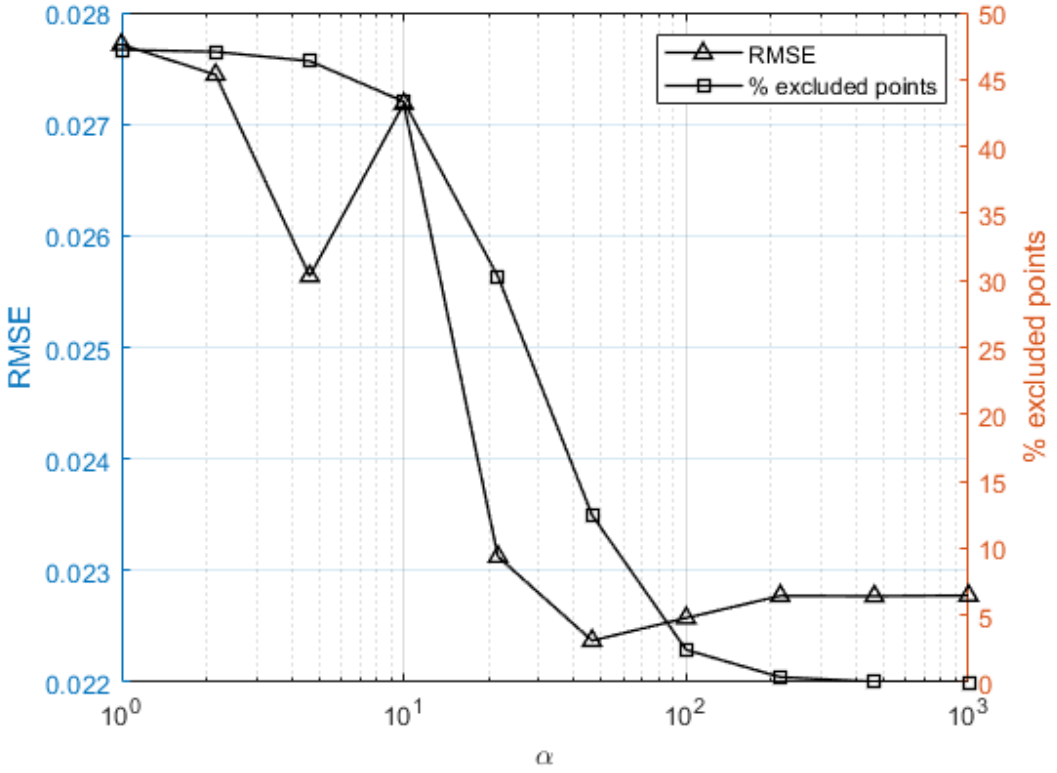


Figure 6.29. Parameter search for ADPE-a using Lasso for SRUb.

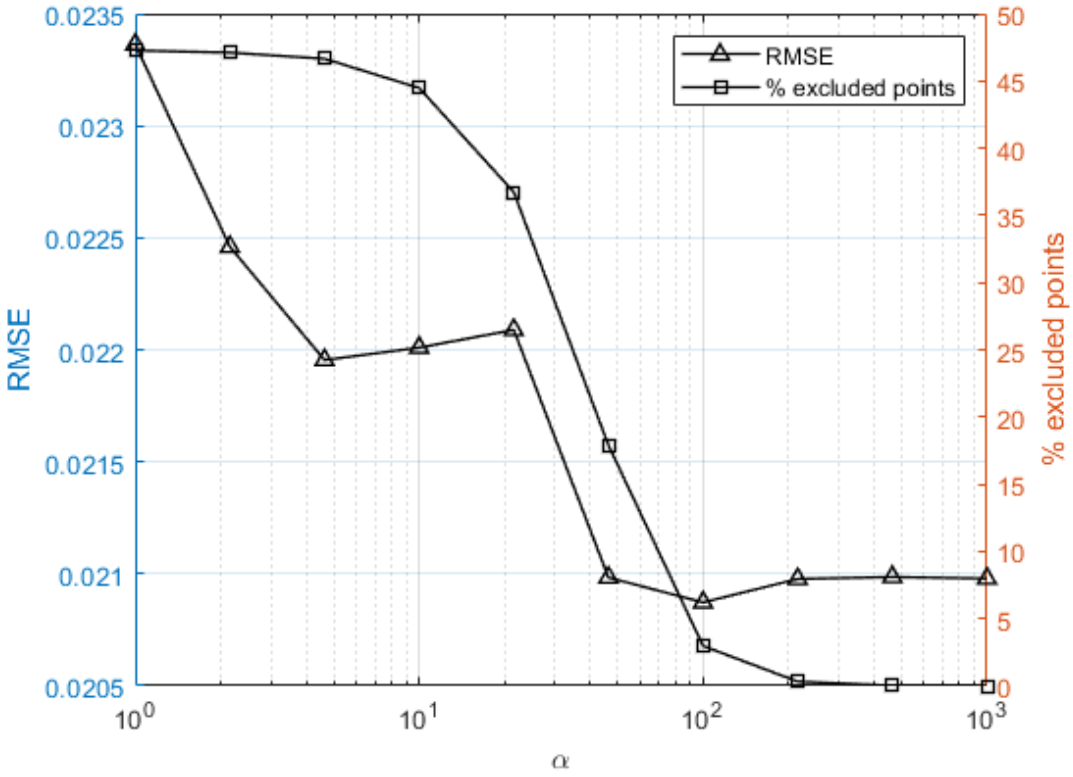


Figure 6.30. Parameter search for ADPE-a using LSSVR for SRUb.

6.2. Parameter Tuning in ADPE-b

ADPE-v consists of two parameters (θ and α in Figure 4.6). Details of the application on on ART1, SRUa and SRUb datasets are given in Appendix A5-A6, A7-A8, A9-A10, respectively. Application on BNZ1 and WWTP datasets are presented below.

6.2.1. ADPE-b on BNZ1 Dataset

ADPE-b is examined on Figure 6.31 for two different constant threshold values for Lasso. If the region where all instances are rejected is taken into account, increasing α has a positive effect on prediction quality. The effect of threshold seems insignificant as the only difference between 2 threshold values is % excluded points, which directly affect RMSE. Hence, large α can be chosen to apply ADPE-b on BNZ1.

It is difficult to catch transition on % excluded points with grid search when LSSVR is used for BNZ1. Hence, Figure 6.32 only shows when θ is 0.12. For LSSVR, increasing α makes % excluded points smaller. RMSE decreases in average and there are points where predictions quite successful even if 35% or 45% of instances are rejected. There is a very sharp decrease in RMSE when α is around 20. There is also an increase in RMSE although % excluded points decreases from 40% to 20% around α is 200 and % excluded points continue to decrease after α is 200. Hence, α between 20 and 50 is an optimum parameter value for ADPE-b when LSSVR is used.

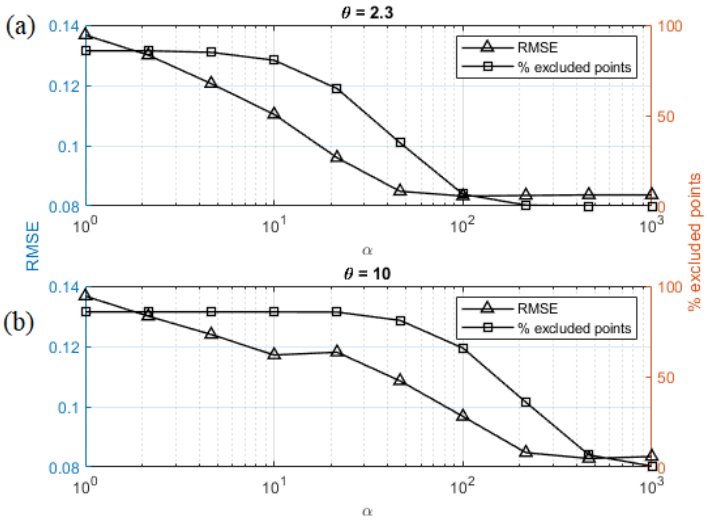


Figure 6.31. Parameter search for ADPE-b with respect to α using Lasso for BNZ1 for (a) $\theta = 0.23$ and (b) θ=1.5.

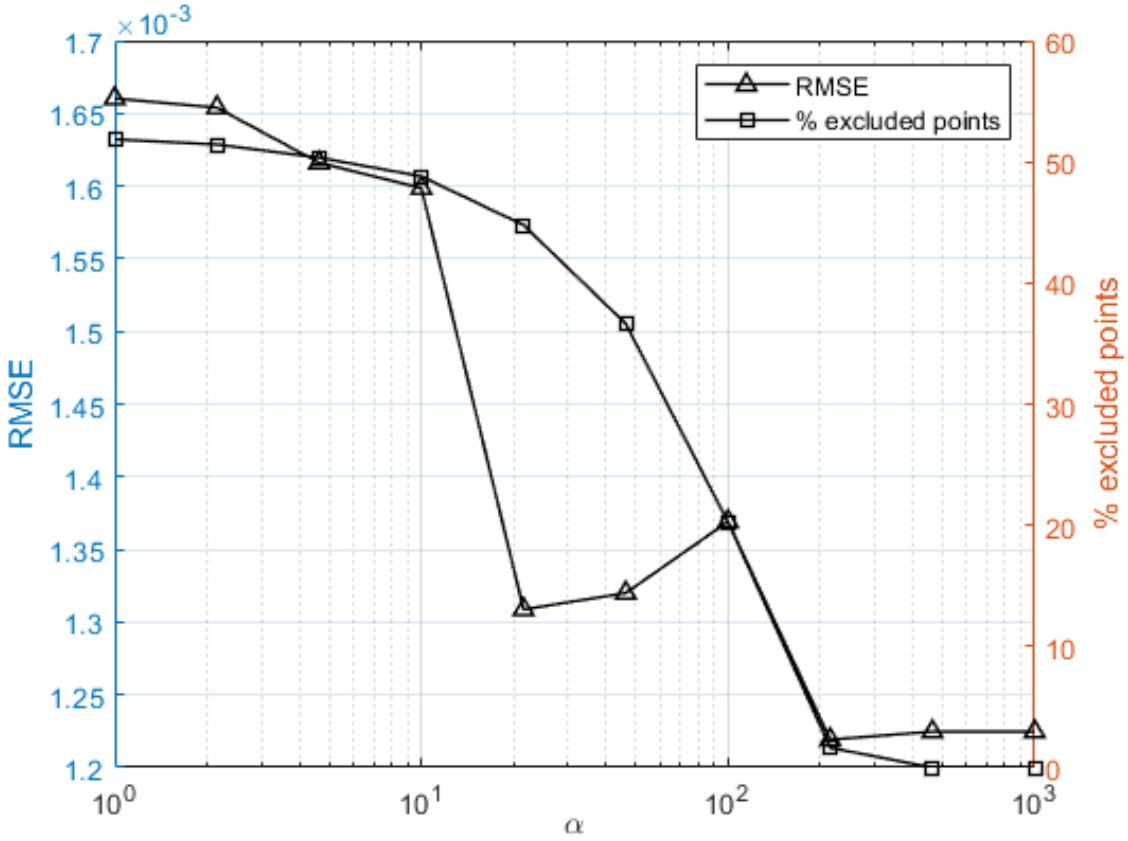


Figure 6.32. Parameter search for ADPE-b with respect to α for BNZ1

6.2.2. ADPE-b on WWTP Dataset

According to Figure 6.33, even if all points are rejected, it is very important which point in the training set is replaced with. When θ is 2.3, as α increases prediction error also decreases in the full rejection region. When α is around 40, nearly 80% of the points are rejected but RMSE is very close to RMSE of JITL. Hence, the increasing effect of prediction error has a positive effect on the prediction quality. When θ is 10, RMSE decreases again as α increases. However, RMSE is higher when it is compared with θ is 2.3. Hence, it can be concluded that large α and moderate θ are suitable for ADPE-b.

The performance of ADPE-b is examined in Figure 6.34. Lasso has a better predictive ability for some instances that are rejected in DMI method and AD. LSSVR is affected more again when ADPE-b is used. RMSE is below 0.025 only when α is greater than 200 for θ is 10. Results are significantly better when the threshold is fixed to 2.3. Hence,

large α and moderate θ parameters are also suitable for LSSVR for WWTP dataset. However, Lasso works better for using data reduction methods in this dataset.

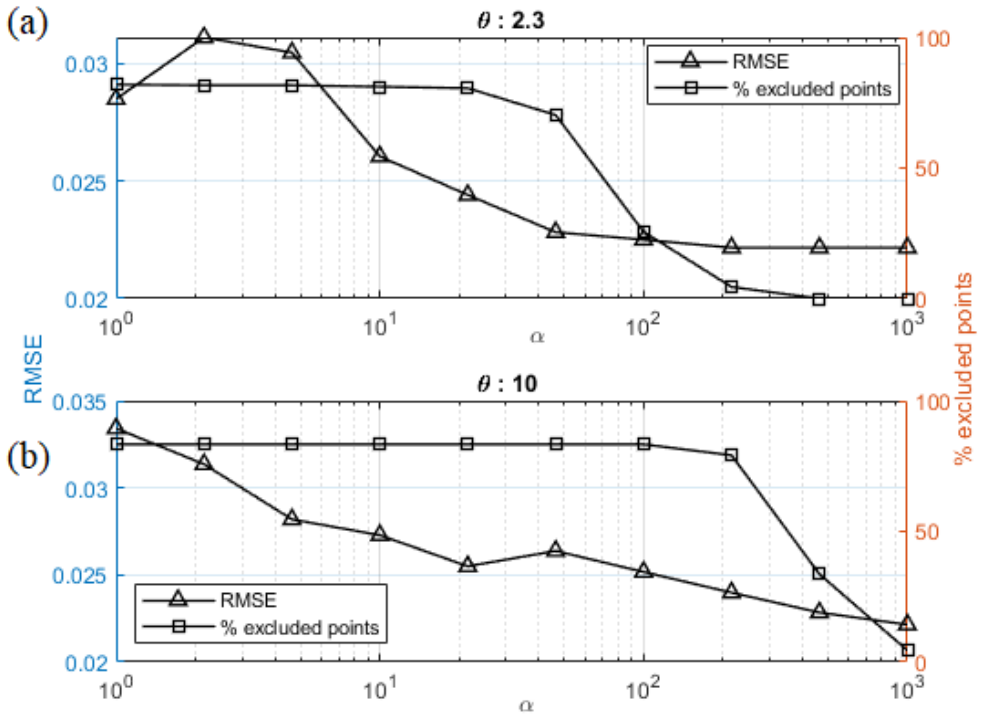


Figure 6.33. Parameter search for ADPE-b with respect to α using Lasso for WWTP for (a) $\theta = 0.23$ and (b) θ=1.5.

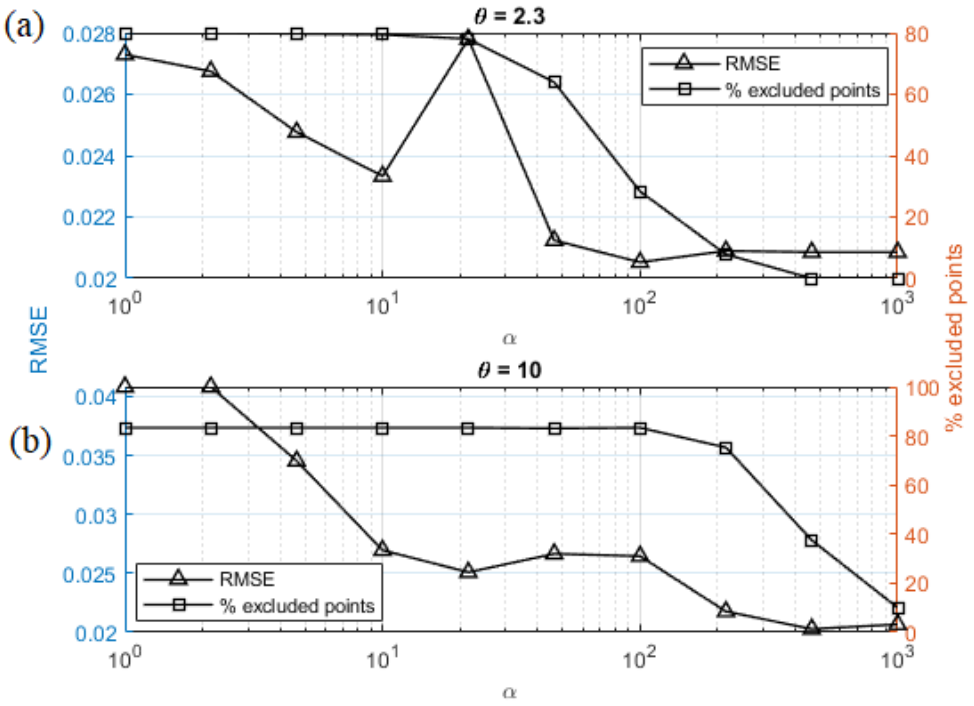


Figure 6.34. Parameter search for ADPE-b with respect to α using LSSVR for WWTP for (a) $\theta = 0.23$ and (b) θ=1.5.

6.3. Comparison of Prediction Accuracy of Data Reduction Methods

After determining the optimum parameters for data reduction methods, results obtained at different % excluded points are smoothened by weighted linear least squares and interpolated at 5% intervals of excluded data points. Relative Additional Error (RAE) is calculated at each interpolated result to determine the prediction performance of the method; RAE gives a normalized prediction accuracy with respect to RMSE of JITL, which includes all the data points:

$$\mathrm{RAE} = \underset{\%\text{exluded point}}{INTP}\left(LOWESS\left(\frac{RMSE - RMSE_{JITL}}{RMSE_{JITL}} \times 100\right)\right) \qquad (6.1)$$

Results for ART1 using Lasso is presented in Figure 6.35, in which each bar with different color corresponds to RAE value of one of the four data reduction methods. Here, ADPE-b, ADPE-a, and AD diverge from DMI method in terms of RAE trend. RAE of DMI method is positive and it increases with % excluded points whereas proposed models show negative RAE values. It should be noted that a negative RAE value is desirable for prediction purposes, since it shows that prediction upon employing the data reduction method improves the prediction accuracy. ADPE-b is also better than ADPE-a and AD after 35% excluded points. Proposed models are effective on ART1 as they always show negative RAE when linear model is used. Figure 6.36 shows the results for ART1 when LSSVR is employed. ADPE-b gives negative RAE between 15% and 40% excluded points. However, the worst case is 2.2% RAE for ADPE-a. Hence, all data reduction methods can be regarded to be successful when they are applied to ART1 with LSSVR. Results for ART2 are given in Appendix B.3 and B.4.

Figure 6.37 shows the results for BNZ1 using Lasso. RAE of ADPE-a, ADPE-b and DMI methods is close to 0 up to 20% excluded points. In general, AD has the worse prediction performance than the other three methods, and prediction accuracy of ADPE-b is also low for 45% and 50% excluded points. RAE of proposed models is under 5% till 30% excluded points. Prediction accuracy of DMI is worse than that of ADPE-b up to 40% excluded points, but gets better between 40% and 50%. Here, the best method seems to be ADPE-a for all excluded points except 50%.

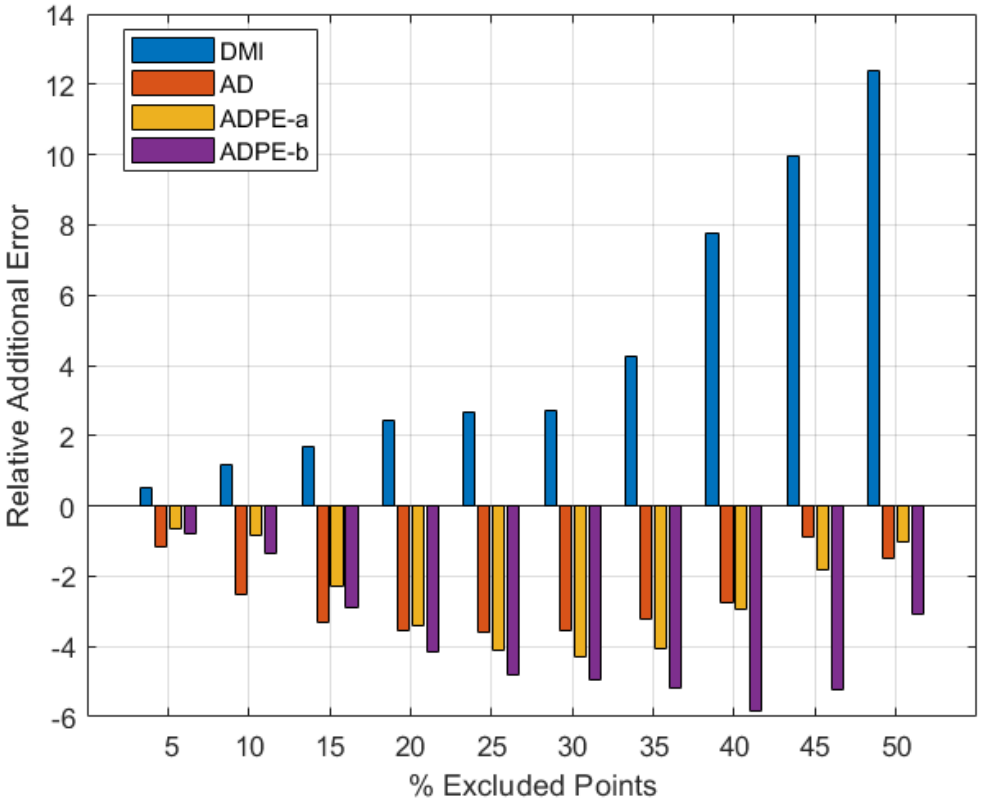


Figure 6.35. Results for ART1 predicted by Lasso.

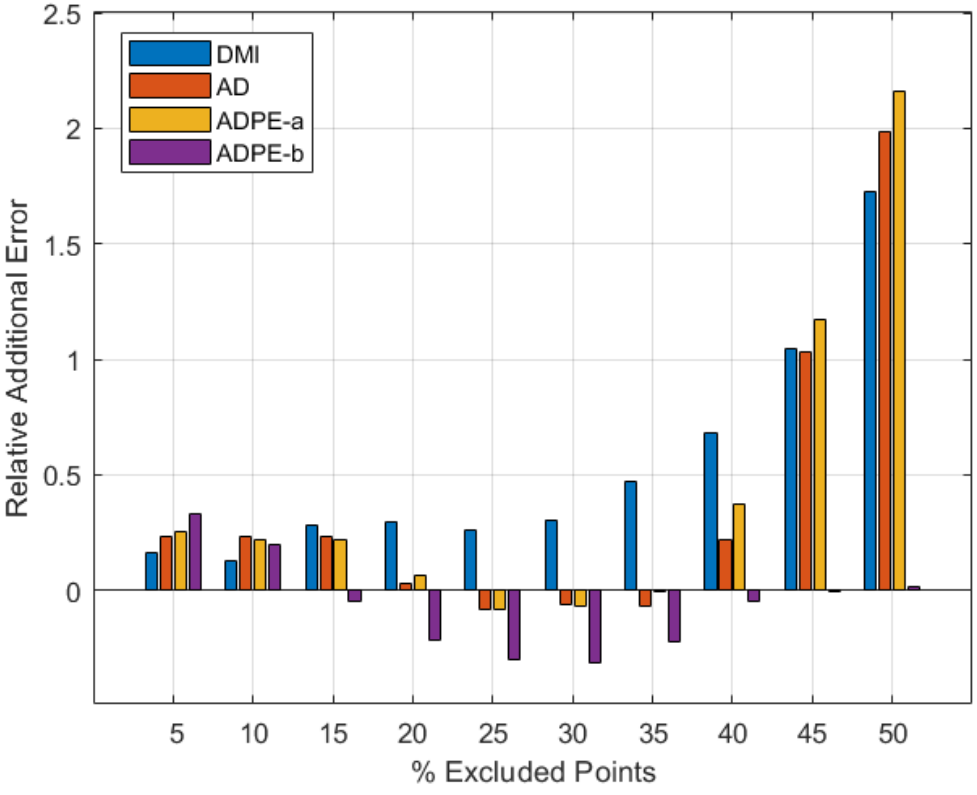


Figure 6.36. Results for ART1 predicted by LSSVR.

Here, the best method seems to be ADPE-a for all excluded points except 50%. Data reduction methods affect the prediction performance of LSSVR in a similar way for BNZ1 (Figure 6.38). The only difference is that proposed models are better than DMI method up to 15% excluded points, and RAE is under 5% in this region. After that point, all four algorithms show similar results, slightly favoring DMI. RAE increases with % excluded points for all methods. Results for BNZ2 are given in Appendix B.3 and B.4.

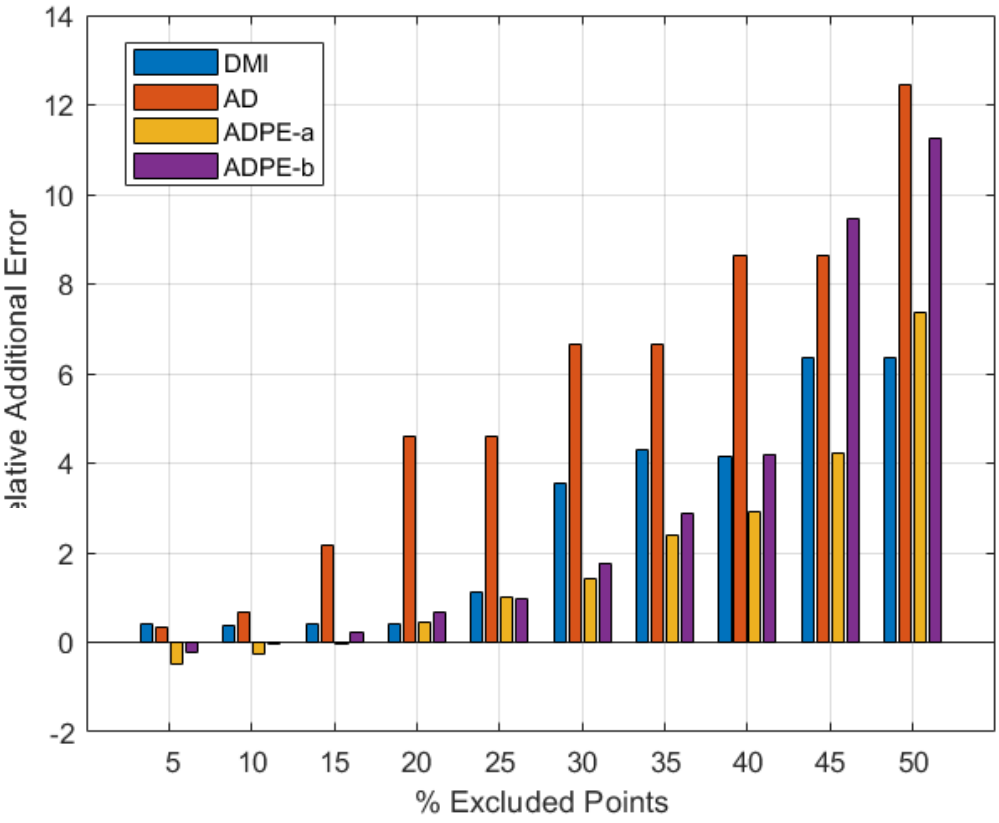


Figure 6.37. Results for BNZ1 predicted by Lasso.

SRUa results for Lasso are shown in Figure 6.39. All four methods are successful up to 30% excluded points, and ADPE-b is the most successful algorithm. RAE of ADPE-b and DMI method is still negative at 30% excluded points. In addition, 8% RAE at 40% excluded points for ADPE-b, and 12% RAE at 50% excluded points are also to be noted. Figure 6.40 shows results for LSSVR. AD is the most successful method up to 30% excluded points. RAE of AD and ADPE-a is negative at 50% excluded points, and ADPE-a is the most robust and successful method in general. Its RAE never exceeds 5% excluded points.

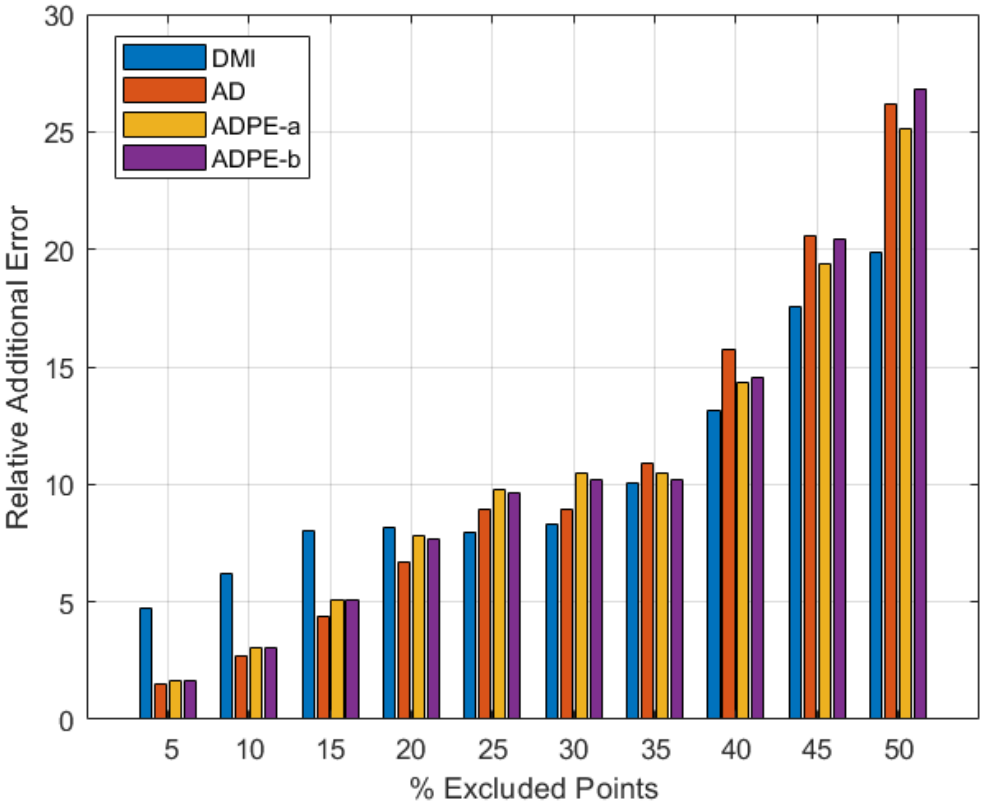


Figure 6.38. Results for BNZ1 predicted by LSSVR.

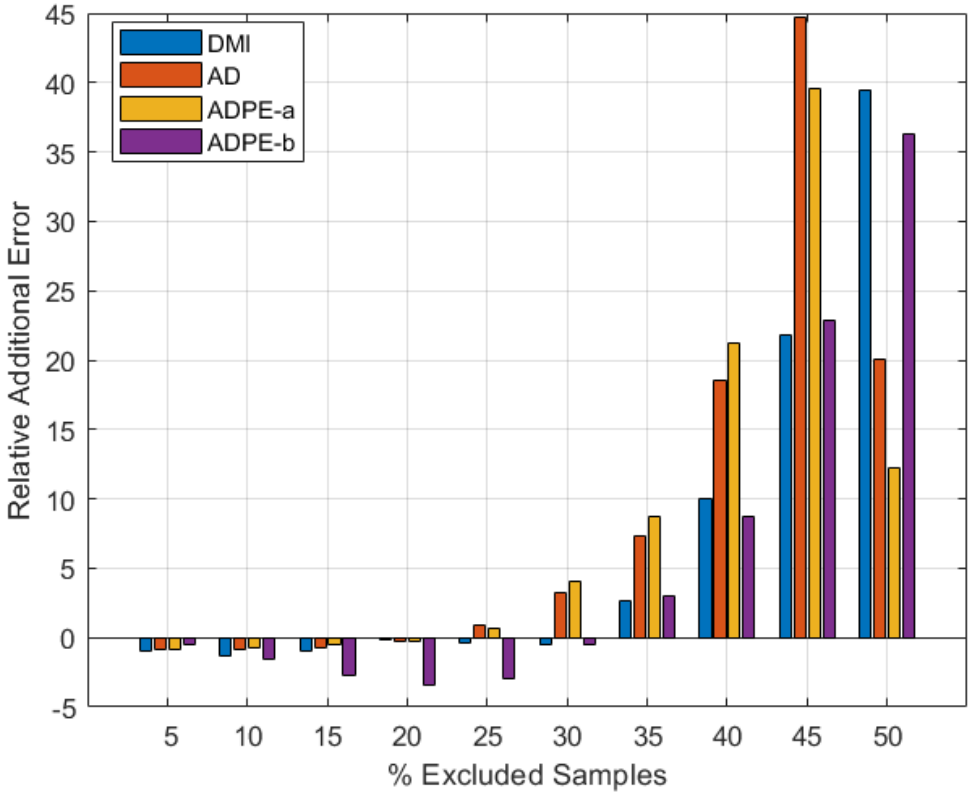


Figure 6.39. Results for SRUa predicted by Lasso.

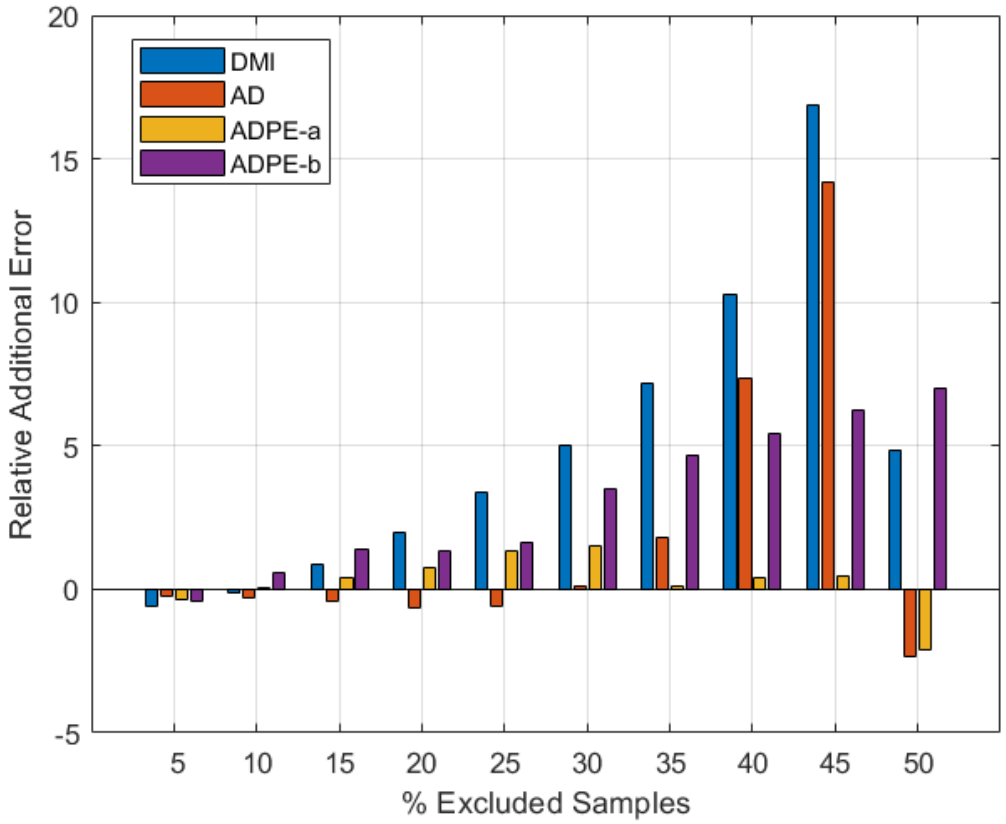


Figure 6.40. Results for SRUa predicted by LSSVR.

Figure 6.42 shows the results for SRUb using Lasso. RAE is around zero for all methods up to 25% excluded points. Hence, it can be said that data reduction is quite effective on SRUb. Proposed algorithms ADPE-a, ADPE-b and AD are especially successful against DMI method at 50% exclude points. Figure 6.43 shows performance of LSSVR for SRUb. All methods are very effective when LSSVR is used. After 40% excluded points, ADPE-a, ADPE-b and AD have better performance than DMI method; RAE value of ADPE-a is lower than -10% at 50% excluded points.

Figure 6.43 shows the results for WWTP using Lasso. It is seen that data reduction is effective on WWTP data. ADPE-b seems the most effective method by giving RAE below 2% at 50% excluded points. It is even possible to take obtain RAE for ADPE-b, ADPE-a and AD till 20% excluded points. DMI method and proposed algorithms show distinct patterns when LSSVR is used for WWTP data, as Figure 6.44 shows. ADPE-b has an increasing prediction performance as % excluded points increases and it is significantly

different from other algorithms, as it gives negative RAE. AD and ADPE-a show similar results whereas DMI method is the worst among all methods.

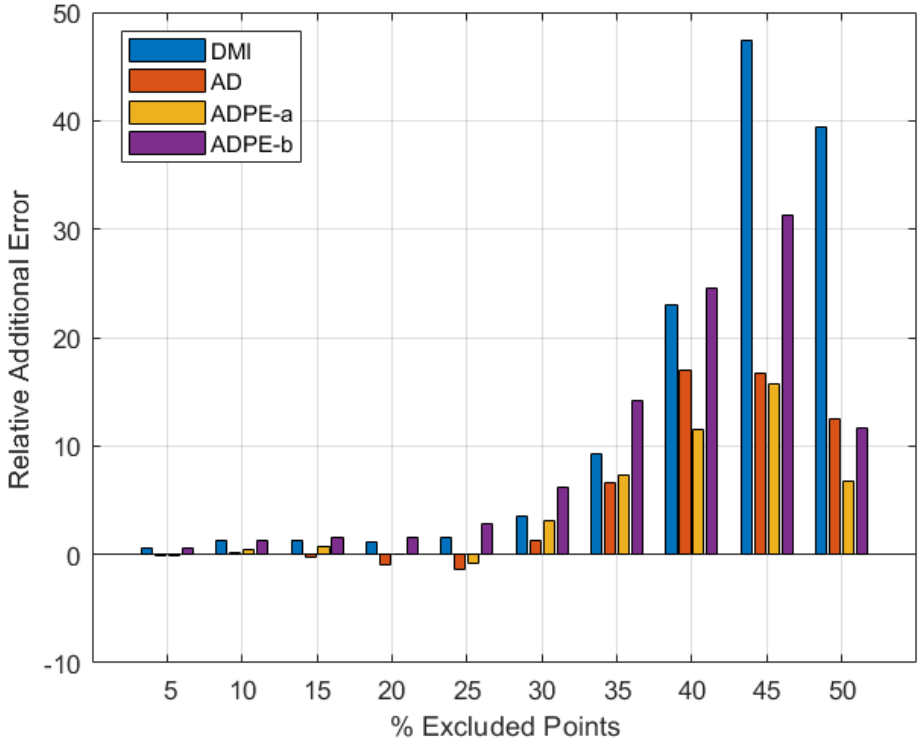


Figure 6.41. Results for SRUb predicted by Lasso.

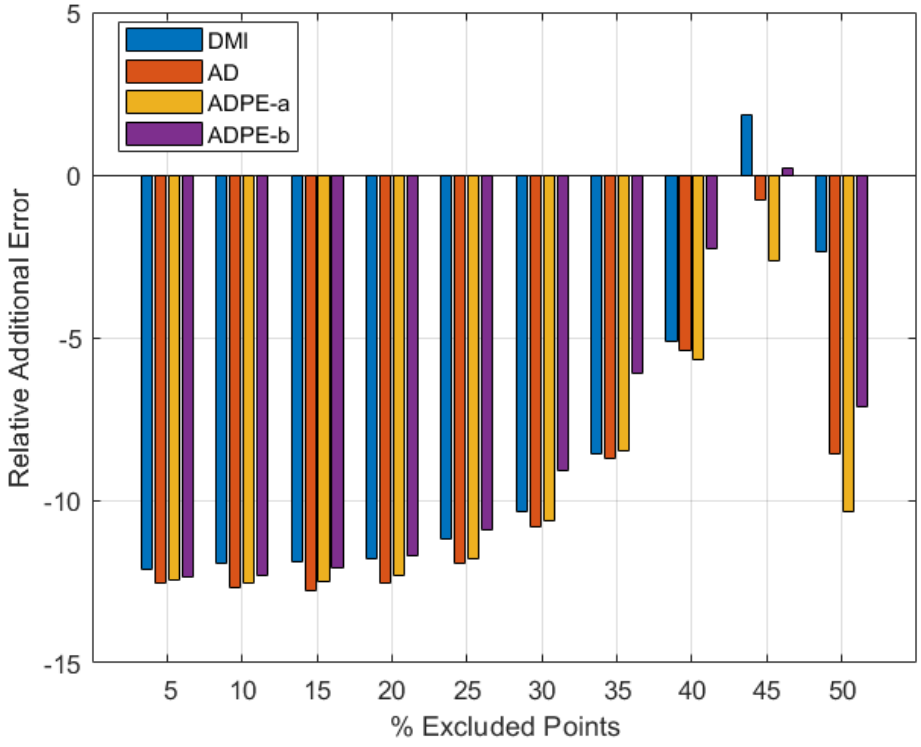


Figure 6.42. Results for SRUb predicted by LSSVR.

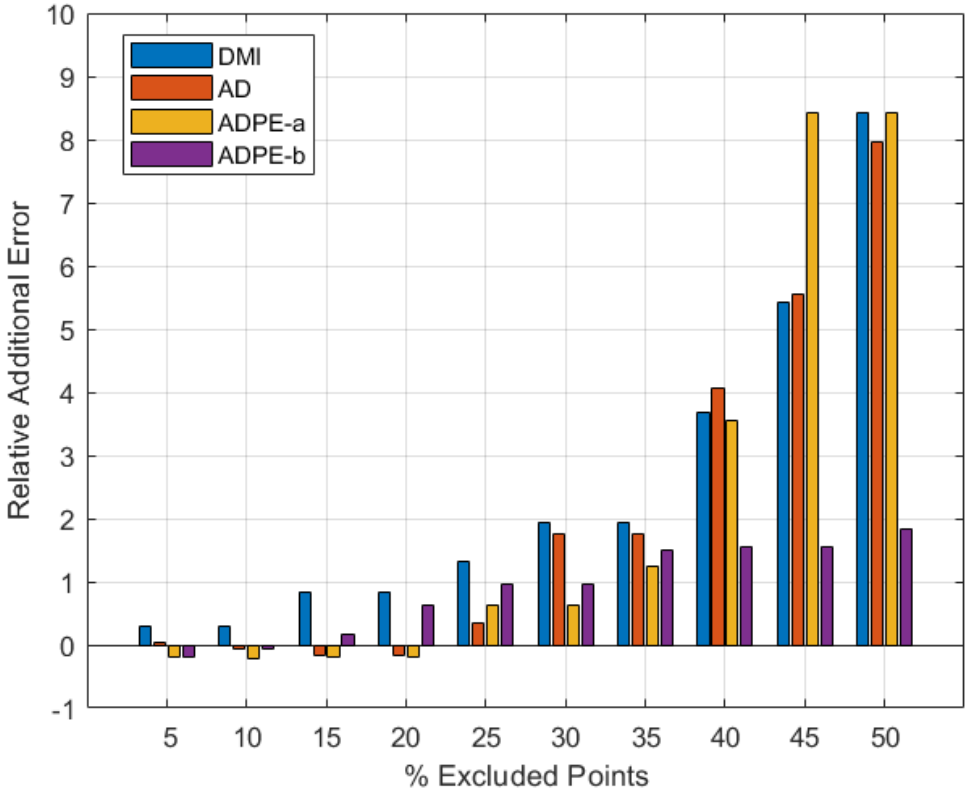


Figure 6.43. Results for WWTP predicted by Lasso.

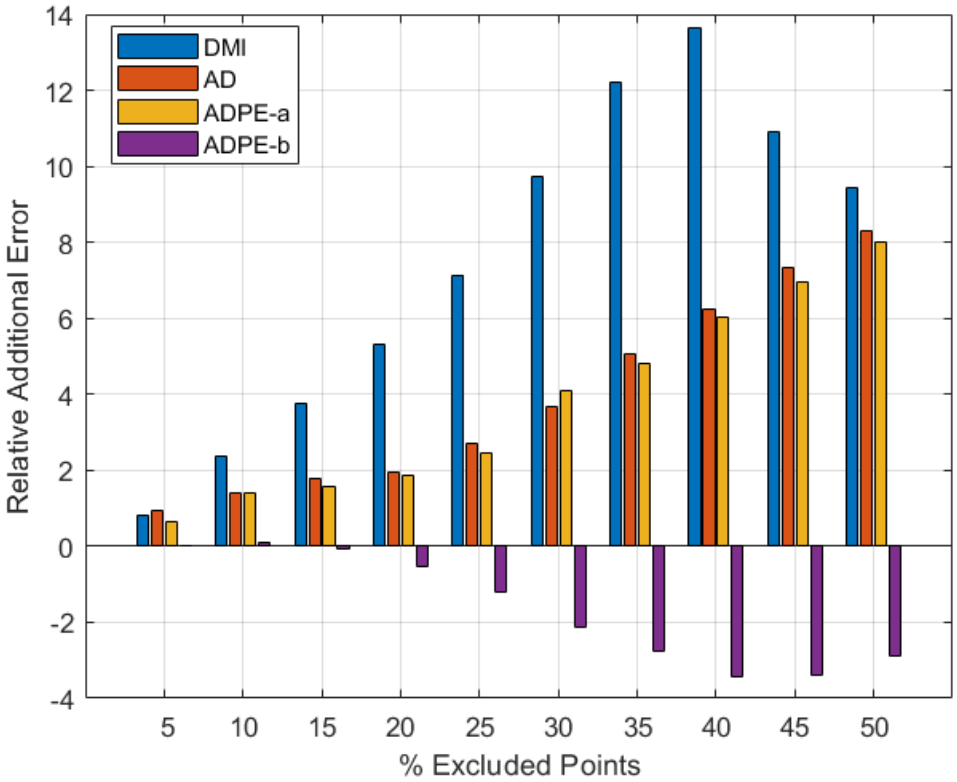


Figure 6.44. Results for WWTP predicted by LSSVR.

7. CONCLUSION

Maintaining high product quality is crucial for industries in a competitive market. Process safety, environmental legislations, and production regulations are important parameters that must be tracked instantaneously to maintain high standards. Hence, it is important to have information about quality variables that indicate the process conditions. These quality variables are often difficult to measure online due to requirement of a time-consuming complex laboratory analysis. Soft sensors provide instant predictions for process monitoring and process control. Prediction accuracy and efficient training algorithms are critical for soft sensing. Instance-based local model, i.e. JITL, has a higher predictive accuracy compared to batch learning via adopting concept drifts and process nonlinearities. On the other hand, storing all the historical data and building a novel model for each query point may computationally be costly. Hence, it is important to keep stored training data as small as possible. Data reduction methods can help to remove redundant samples from the database. In the current thesis, four different data reduction methods are applied using Lasso and LSSVR learners on four different (one synthetic and three industrial) datasets.

In all dataset, LSSVR is found to have a better prediction performance than Lasso using JITL, and BNZ and ART datasets show that Lasso benefits more from large training sets compared to LSSVR. This shows that soft sensors should be constructed using LSSVR instead of linear learners, particularly in the absence of a large historical database.

Four different data reduction methods are applied: DMI method, AD, ADPE-a and ADPE-b. DMI method from the literature is used to compare proposed algorithms. The aim is to keep database size as small as possible while keeping predictive ability better or close to performance of JITL. Parameters of data reduction methods are optimized through grid search, and their sensitivity, i.e. effects on the prediction accuracy, is determined. It is seen that accuracy of DMI method is highly sensitive to its two parameters, and the optimum results are obtained in a narrow range of parameter values. On the other hand, the optimum prediction accuracy values reached by the proposed data reduction methods are found to be less sensitive to parameter values, i.e. a wider range of parameter values yield RMSE as small as the optimum ones.

It is interesting to note that it is possible to improve prediction accuracy using data reduction methods up to ~25% excluded points both LSSVR Lasso and SVR learners. While prediction accuracy improvement seems to be more common feature of Lasso employed on various datasets, application of data reduction methods using LSSVR on on SRUb dataset improves prediction accuracy even for 50% excluded points when. This shows that utility of data reduction methods may extend beyond database management capability and may be utilized as tools to improve prediction accuracy of JITL methods. One explanation of why prediction accuracy is increased may be the constant number of nearest neighbors used: As the density of the data space is increased, a constant NN size may be spanning a smaller volume, hence degrading the prediction accuracy. Adjusting the data density using data reduction methods may be helpful in maintaining predictive accuracy using the constant NN size.

Between 25-40% excluded points, prediction accuracy may deteriorate ~10% compared to JITL upon using data reduction methods. This shows that data reduction methods may be carefully employed to exclude an upper limit of 40% data points, but a higher value may significantly deteriorate the prediction accuracy. In other words, keeping a constant size database may not be a convenient method for obtaining high prediction accuracy, while a slow increase in database size should be manageable, maintaining a desirable prediction accuracy. Up to ~40% excluded points, Lasso seems more immune to deterioration in prediction accuracy, compared to LSSVR, while deterioration in LSSVR predictions is less severe compared to Lasso prediction for 40-50% excluded points.

It is seen that the proposed three methods generally yield superior prediction accuracy compared to DMI method. The most drastic difference is seen for <~30% excluded points, which also correspond to a negligible deterioration in prediction accuracy. Here, ADPE-a generally seems to be the best method yielding smallest prediction error values. Furthermore, increasing up to 50% excluded points using ADPE-a may still yield good predicti on accuracy, showing that this method may be used for database management, and yield high prediction accuracy for both linear and nonlinear learners; using ADPE-a up to ~30% excluded points is likely to increase prediction accuracy compared to a sole JITL method.

REFERENCES

1. Soft Sensors in Industrial Applications, pp. 1–13, Springer London, London, 2007.

2. Souza, F. A., R. Araujo and J. Mendes, “Review of soft sensor methods for regression applications”, Chemometrics and Intelligent Laboratory Systems, Vol. 152, pp. 69–79, 2016.

3. Rogina, A., Šiško, I., Mohler, I., Ujević, Ž. and Bolf, N., 2011. Soft sensor for continuous product quality estimation (in crude distillation unit). Chemical Engineering Research and Design, 89(10), pp.2070-2077.

4. Kadlec, P., B. Gabrys and S. Strandt, “Data-driven soft sensors in the process industry”, Computers & chemical engineering, Vol. 33, No. 4, pp. 795–814, 2009.

5. Arnaiz-González, Á., Blachnik, M., Kordos, M. and García-Osorio, C., 2016. Fusion of instance selection methods in regression tasks. Information Fusion, 30, pp.69-79.

6. Antonelli, M., P. Ducange and F. Marcelloni, “Genetic training instance selection in multiobjective evolutionary fuzzy systems: A coevolutionary approach”, IEEE Transactions on fuzzy systems, Vol. 20, No. 2, pp. 276–290, 2011.

7. Olvera-López, J., Carrasco-Ochoa, J., Martínez-Trinidad, J. and Kittler, J., 2010. A review of instance selection methods. Artificial Intelligence Review, 34(2), pp.133-143.

8. James, G., D. Witten, T. Hastie and R. Tibshirani, An introduction to statistical learning, Vol. 112, Springer, 2013.

9. Hawkins, D. M., “The problem of overfitting”, Journal of chemical information and computer sciences, Vol. 44, No. 1, pp. 1–12, 2004.

10. Montgomery, D. C., E. A. Peck and G. G. Vining, Introduction to linear regression analysis, Vol. 821, John Wiley & Sons, 2012.

11. Geladi, P. and B. R. Kowalski, “Partial least-squares regression: a tutorial”, Analytica chimica acta, Vol. 185, pp. 1–17, 1986.

12. Haenlein, M. and A. M. Kaplan, “A beginner’s guide to partial least squares analysis”, Understanding statistics, Vol. 3, No. 4, pp. 283–297, 2004.

13. Vinzi, V. E., W. W. Chin, J. Henseler, H. Wang et al., Handbook of partial least squares, Vol. 201, Springer, 2010.

14. Wold, S., Sjöström, M. and Eriksson, L., 2001. PLS-regression: a basic tool of chemometrics. Chemometrics and Intelligent Laboratory Systems, 58(2), pp.109-130.

15. Kyung, M., J. Gill, M. Ghosh, G. Casella et al., “Penalized regression, standard errors, and Bayesian lassos”, Bayesian Analysis, Vol. 5, No. 2, pp. 369–411, 2010

16. Hoerl, A. E. and R. W. Kennard, “Ridge regression: applications to nonorthogonal problems”, Technometrics, Vol. 12, No. 1, pp. 69–82, 1970.

17. Marquaridt, D. W., “Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation”, Technometrics, Vol. 12, No. 3, pp. 591–612, 1970.

18. Tibshirani, R., “Regression shrinkage and selection via the lasso”, Journal of the Royal Statistical Society: Series B (Methodological), Vol. 58, No. 1, pp. 267–288, 1996.

19. Tibshirani, R., “The lasso method for variable selection in the Cox model”, Statistics in medicine, Vol. 16, No. 4, pp. 385–395, 1997.

20. Drucker, H., C. J. Burges, L. Kaufman, A. J. Smola and V. Vapnik, “Support vector regression machines”, Advances in neural information processing systems, pp. 155–161, 1997.

21. De Brabanter, J., B. De Moor, J. A. Suykens, T. Van Gestel and J. P. Vandewalle, Least squares support vector machines, World scientific, 2002.

22. Cortes, C. and V. Vapnik, “Support-vector networks”, Machine learning, Vol. 20, No. 3, pp. 273–297, 1995.

23. Saunders, C., A. Gammerman and V. Vovk, “Ridge regression learning algorithm in dual variables”, 1998.

24. Cawley, G. C. and N. L. Talbot, “Reduced rank kernel ridge regression”, Neural Processing Letters, Vol. 16, No. 3, pp. 293–302, 2002.

25. Ge, Z., Song, Z., Ding, S. and Huang, B., 2017. Data Mining and Analytics in the Process Industry: The Role of Machine Learning. IEEE Access, 5, pp.20590-20616.

26. Ge, Z., Song, Z. and Gao, F., 2013. Review of Recent Research on Data-Based Process Monitoring. Industrial & Engineering Chemistry Research, 52(10), pp.3543-3562.

27. Khatibisepehr, S., Huang, B. and Khare, S., 2013. Design of inferential sensors in the process industry: A review of Bayesian methods. Journal of Process Control, 23(10), pp.1575-1596.

28. Ge, Z., 2017. Review on data-driven modeling and monitoring for plant-wide industrial processes. Chemometrics and Intelligent Laboratory Systems, 171, pp.16

29. Souza, F., Araújo, R. and Mendes, J., 2016. Review of soft sensor methods for regression applications. Chemometrics and Intelligent Laboratory Systems, 152, pp.69-79

30. Lin, B., Recke, B., Knudsen, J. and Jørgensen, S., 2007. A systematic approach for soft sensor development. Computers & Chemical Engineering, 31(5-6), pp.419-425.

31. Kano, M. and Fujiwara, K., 2013. Virtual Sensing Technology in Process Industries: Trends and Challenges Revealed by Recent Industrial Applications. Journal Of Chemical Engineering of Japan, pp.1-17.

32. Kadlec, P., Grbić, R. and Gabrys, B., 2011. Review of adaptation mechanisms for data-driven soft sensors. Computers & Chemical Engineering, 35(1), pp.1-24.

33. Saptoro, A., 2014. State of the Art in the Development of Adaptive Soft Sensors based on Just-in-Time Models. Procedia Chemistry, 9, pp.226-234.

34. Bontempi, G., Birattari, M. and Bersini, H., 1999. Lazy learning for local modelling and control design. International Journal of Control, 72(7-8), pp.643-658

35. Shao, W., Ge, Z. and Song, Z., 2020. Bayesian Just-in-Time Learning and Its Application to Industrial Soft Sensing. IEEE Transactions on Industrial Informatics, 16(4), pp.2787-2798.

36. Liu, Y., Gao, Z., Li, P. and Wang, H., 2012. Just-in-Time Kernel Learning with Adaptive Parameter Selection for Soft Sensor Modeling of Batch Processes. Industrial & Engineering Chemistry Research, 51(11), pp.4313-4327.

37. Mei, C., Chen, Y., Jiang, H., Ding, Y., Chen, X. and Liu, G., 2017. Just-in-time Modeling with a Combination of Input and Output Similarity Criterions for Soft Sensor Modeling in Fermentation Processes. Chemical Engineering Transactions, 61, pp.1045-1050.

38. Yuan, X., Zhou, J., Wang, Y. and Yang, C., 2018. Multi-similarity measurement driven ensemble just-in-time learning for soft sensing of industrial processes. Journal of Chemometrics, 32(9), pp. 3040.

39. Fujiwara, K., M. Kano, S. Hasebe and A. Takinami, “Soft-sensor development using correlation-based just-in-time modeling”, AIChE Journal, Vol. 55, No. 7, pp. 1754–1765, 2009.

40. Zhang, J., Y.-S. Yim and J. Yang, “Intelligent selection of instances for prediction functions in lazy learning algorithms”, Lazy learning, pp. 175–191, Springer, 1997.

41. Nock, R., M. Sebban and D. Bernard, “A simple locally adaptive nearest neighbor rule with application to pollution forecasting”, International journal of pattern recognition and artificial intelligence, Vol. 17, No. 08, pp. 1369–1382, 2003.

42. Yao, L. and Ge, Z., 2017. Moving window adaptive soft sensor for state shifting process based on weighted supervised latent factor analysis. Control Engineering Practice, 61, pp.72-80.

43. Yao, L. and Ge, Z., 2017. Online Updating Soft Sensor Modeling and Industrial Application Based on Selectively Integrated Moving Window Approach. IEEE Transactions on Instrumentation and Measurement, 66(8), pp.1985-1993.

44. Alakent, B., 2020. Soft sensor design using transductive moving window learner. Computers & Chemical Engineering, 140, p.106941.

45. Kneale, C. and Brown, S., 2018. Small moving window calibration models for soft sensing processes with limited history. Chemometrics and Intelligent Laboratory Systems, 183, pp.36-46.

46. Kaneko, H. and Funatsu, K., 2011. Maintenance-free soft sensor models with time difference of process variables. Chemometrics and Intelligent Laboratory Systems, 107(2), pp.312-317.

47. Xiong, W., Li, Y., Zhao, Y. and Huang, B., 2017. Adaptive soft sensor based on time difference Gaussian process regression with local time-delay reconstruction. Chemical Engineering Research and Design, 117, pp.670-680.

48. Kaneko, H. and Funatsu, K., 2011. Development of Soft Sensor Models Based on Time Difference of Process Variables with Accounting for Nonlinear Relationship. Industrial & Engineering Chemistry Research, 50(18), pp.10643-10651.

49. Kaneko, H. and Funatsu, K., 2011. A soft sensor method based on values predicted from multiple intervals of time difference for improvement and estimation of prediction accuracy. Chemometrics and Intelligent Laboratory Systems, 109(2), pp.197-206.

50. Liu, H. and H. Motoda, "On issues of instance selection", Data Mining and Knowledge Discovery, Vol. 6, No. 2, p. 115, 2002.

51. Kaneko, H. and K. Funatsu, "Ensemble locally weighted partial least squares as a just-in-time modeling method", AIChE Journal, Vol. 62, No. 3, pp. 717–725, 2016

52. Hart, P., "The condensed nearest neighbor rule (Corresp.)", IEEE transactions on information theory, Vol. 14, No. 3, pp. 515–516, 1968.

53. Wilson, D. L., "Asymptotic properties of nearest neighbor rules using edited data", IEEE Transactions on Systems, Man, and Cybernetics, No. 3, pp. 408–421, 1972.

54. Barandela, R., F. J. Ferri and J. S. Sanchez, "Decision boundary preserving prototype selection for nearest neighbor classification", International Journal of Pattern Recognition and Artificial Intelligence, Vol. 19, No. 06, pp. 787–806, 2005.

55. Kaneko, H. and K. Funatsu, "Database monitoring index for adaptive soft sensors and the application to industrial process", AIChE Journal, Vol. 60, No. 1, pp. 160–169, 2014

56. De Vito, S., Massera, E., Piga, M., Martinotto, L. and Di Francia, G., 2008. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. Sensors and Actuators B: Chemical, 129(2), pp.750-757.

57. Fortuna, L., Rizzo, A., Sinatra, M. and Xibilia, M., 2003. Soft analyzers for a sulfur recovery unit. Control Engineering Practice, 11(12), pp.1491-1500.

58. Souza, F., Araújo, R., Matias, T. and Mendes, J., 2013. A multilayer-perceptron based method for variable selection in soft sensor design. Journal of Process Control, 23(10), pp.1371-1378.

APPENDIX A: PARAMETER SEARCH FOR DATA REDUCTION METHODS

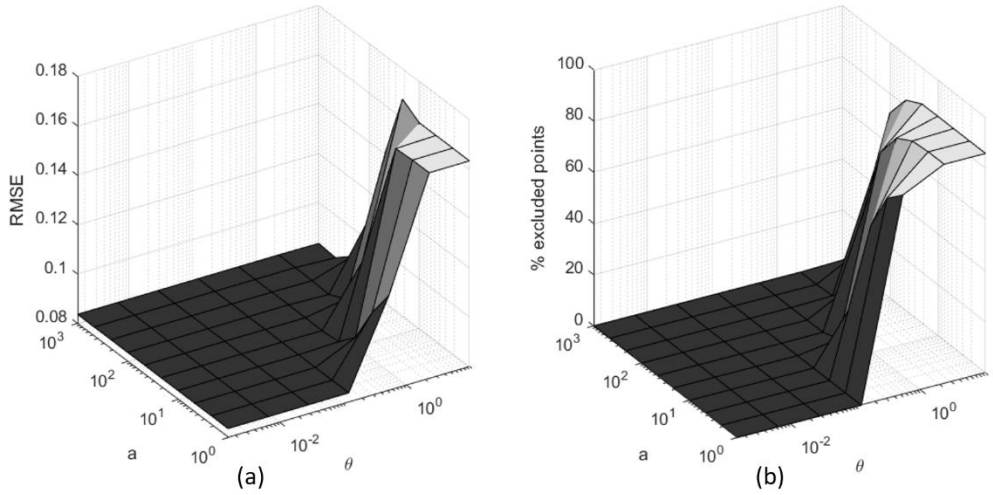


Figure A.1. Parameter search for ADPE-a with respect to (a) θ and (b) α using Lasso for BNZ1.

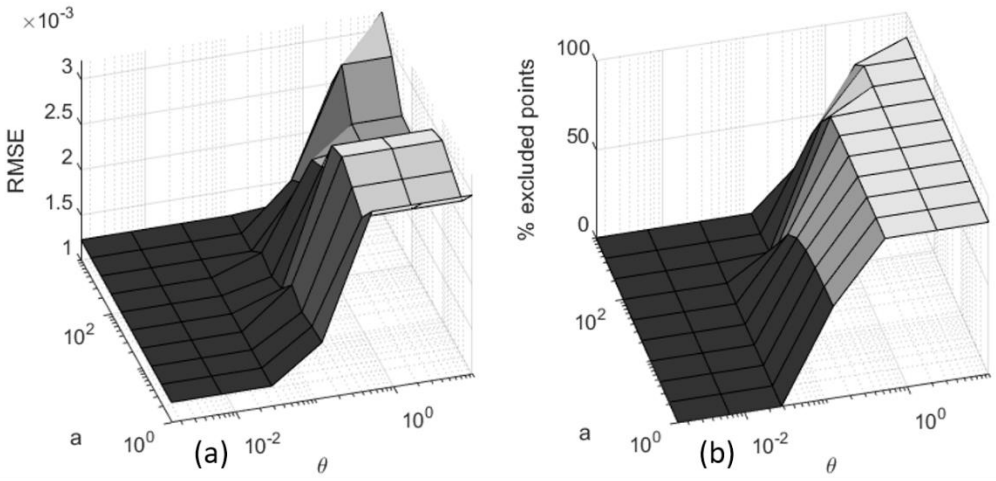


Figure A.2. Parameter search for ADPE-a with respect to (a) θ and (b) α using LSSVR for BNZ1.

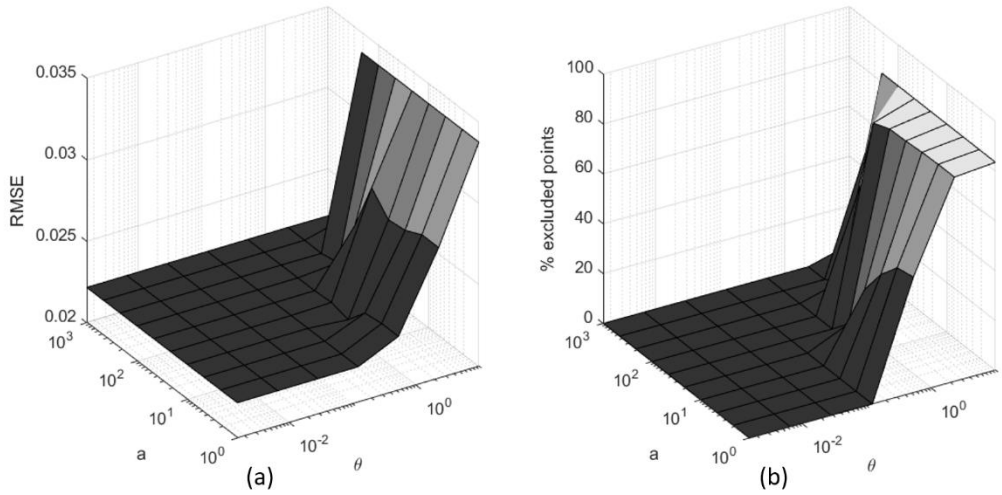


Figure A.3. Parameter search for ADPE-a with respect to (a) θ and (b) α using Lasso for WWTP.

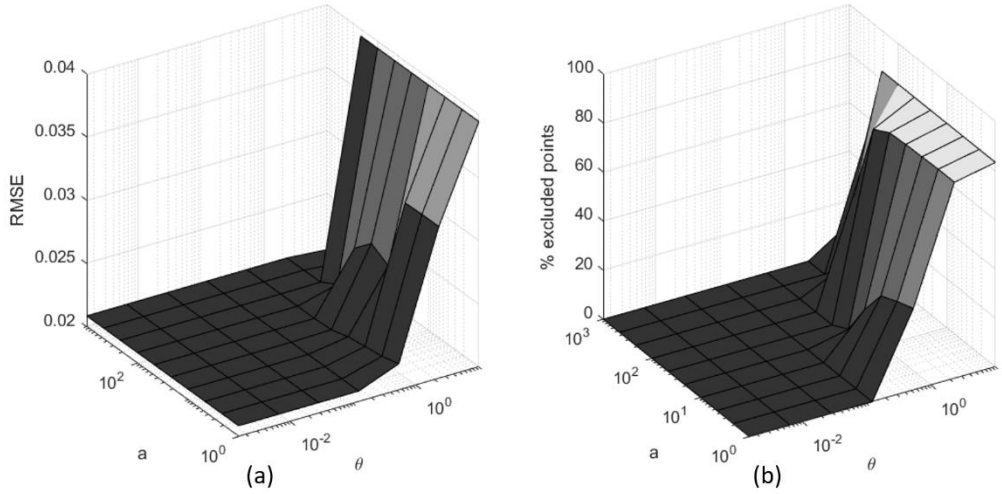


Figure A.4. Parameter search for ADPE-a with respect to (a) θ and (b) α using LSSVR for WWTP.

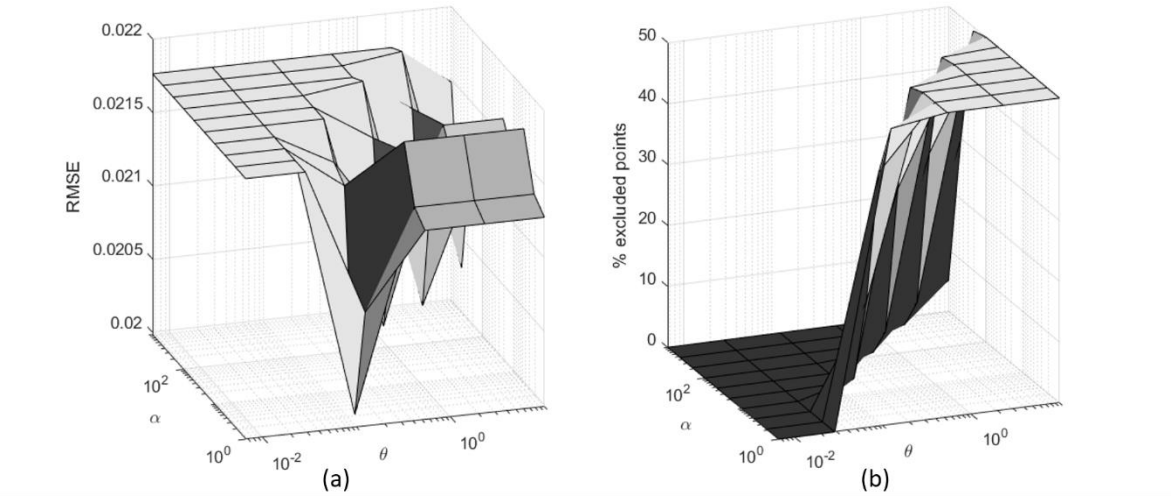


Figure A.5. Parameter search for ADPE-b with respect to (a) θ and (b) α using Lasso for ART1.

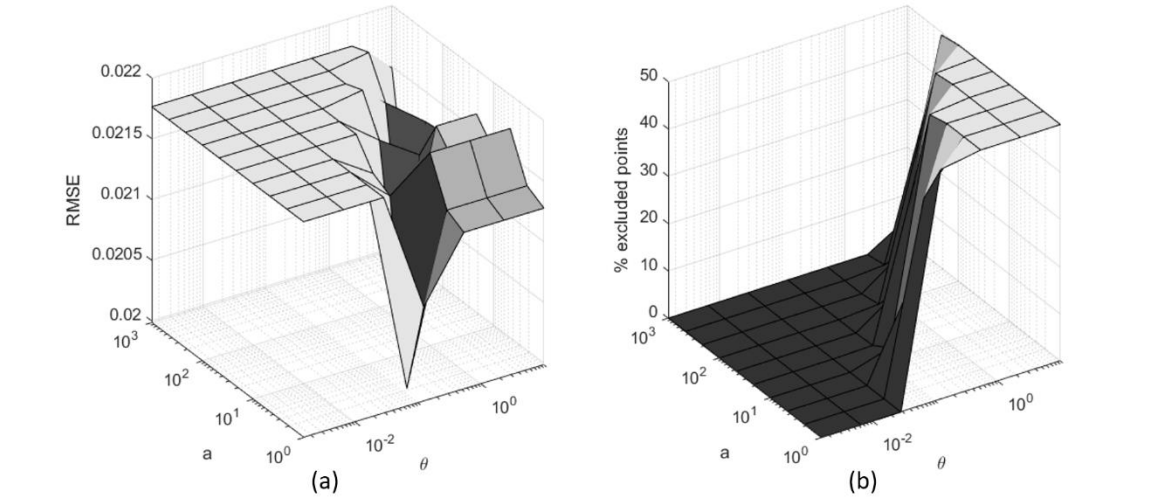


Figure A.6. Parameter search for ADPE-b with respect to (a) θ and (b) α using LSSVR for ART1.

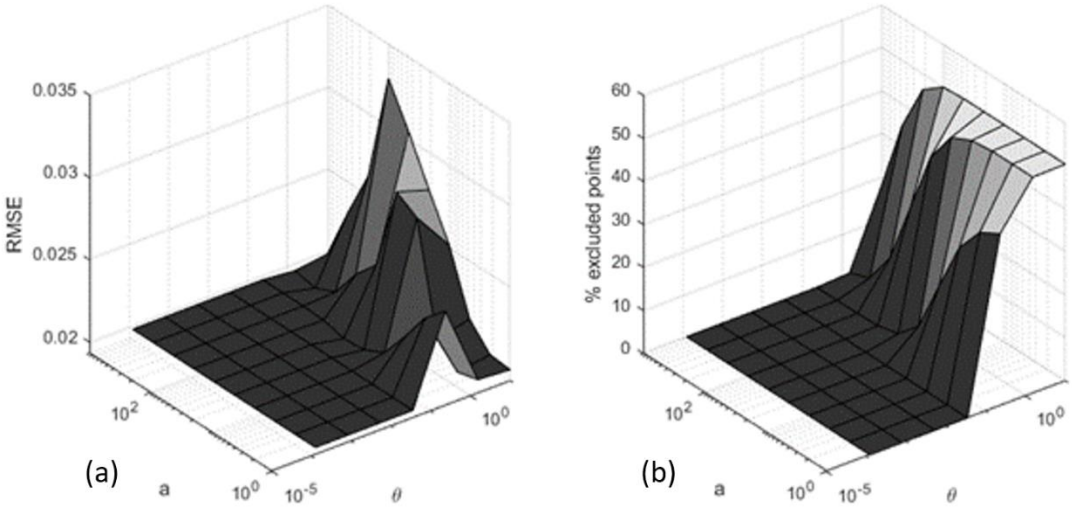


Figure A.7. Parameter search for ADPE-b with respect to (a) θ and (b) α using Lasso for SRUa.

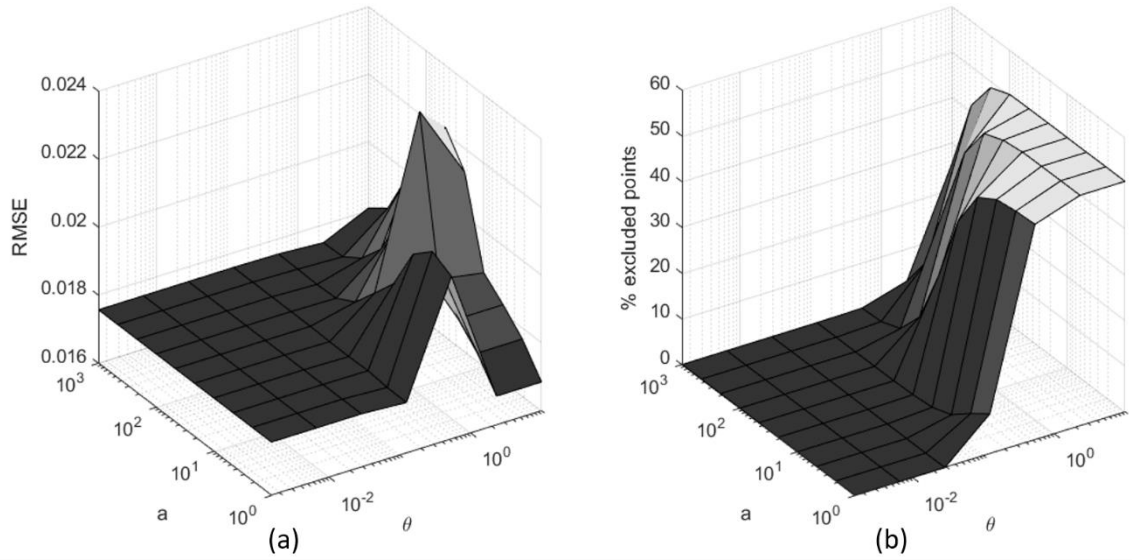


Figure A.8. Parameter search for ADPE-b with respect to (a) θ and (b) α using LSSVR for SRUa.

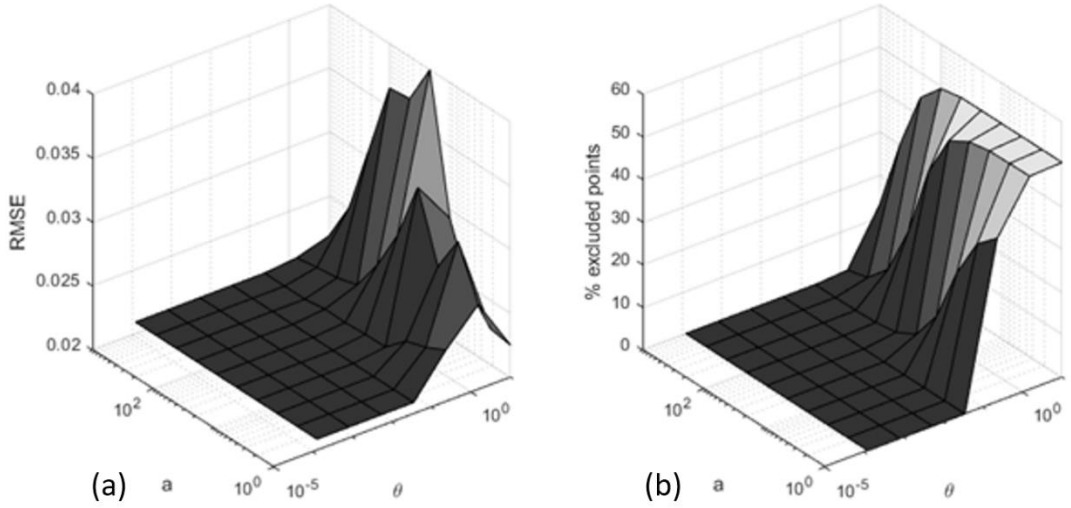


Figure A.9. Parameter search for ADPE-b with respect to (a) θ and (b) α using Lasso for SRUb.

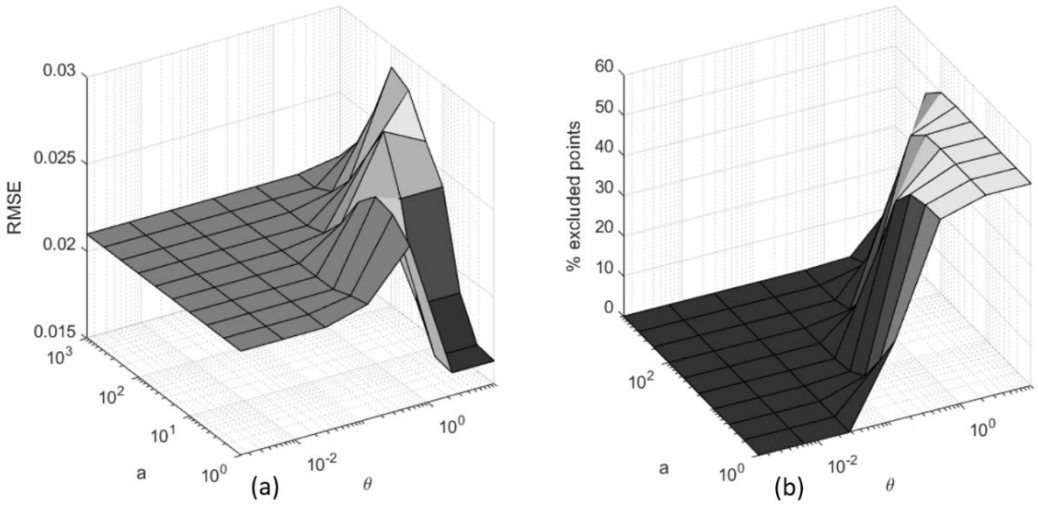


Figure A.10. Parameter search for ADPE-b with respect to (a) θ and (b) α using LSSVR for SRUb.

APPENDIX B: RESULTS FOR DATA REDUCTION METHODS

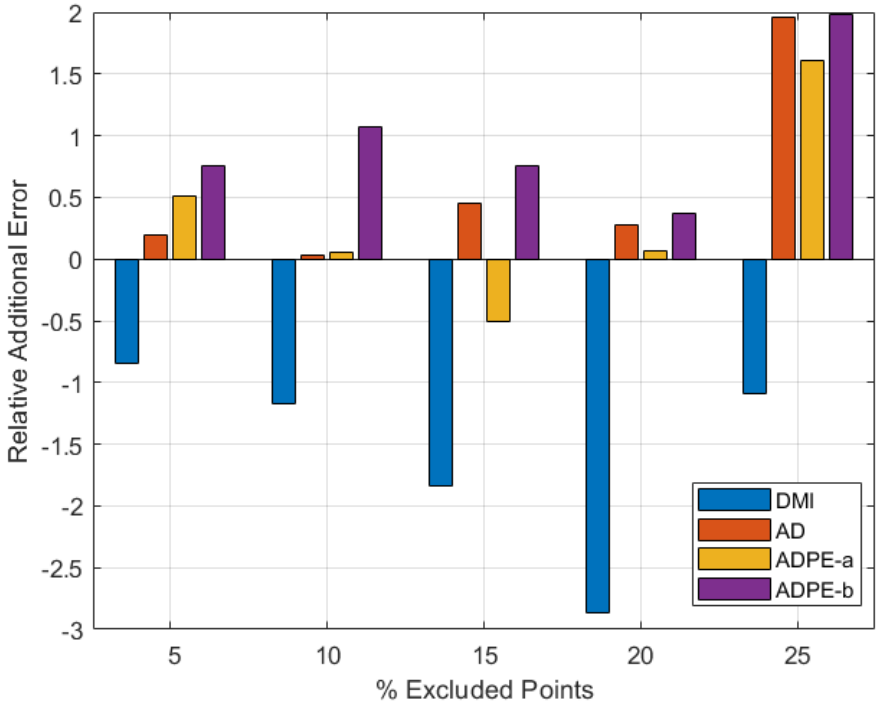


Figure B.1. Results for ART2 predicted by Lasso.

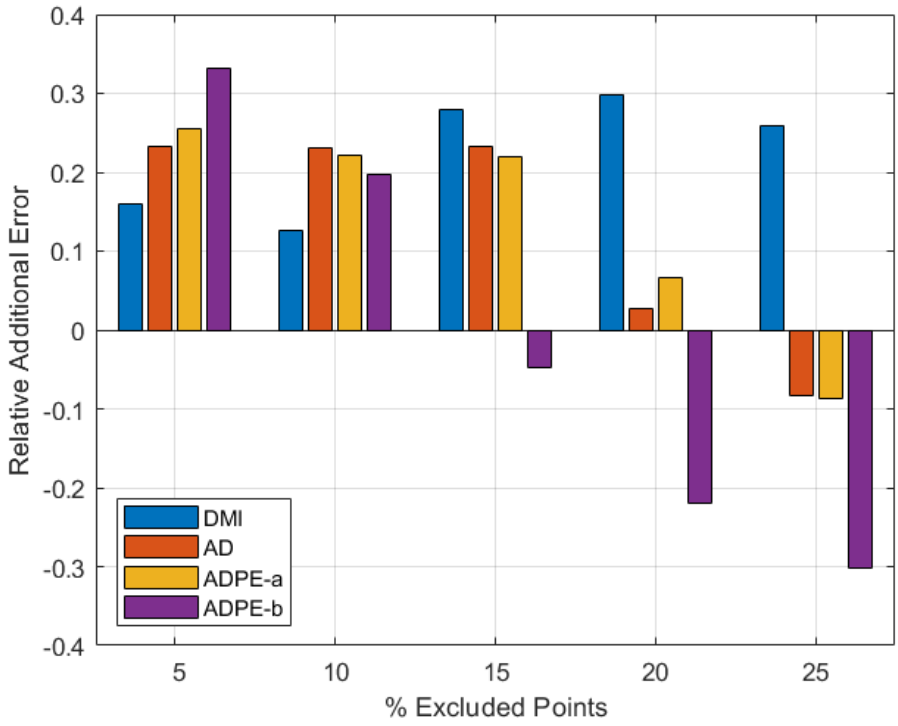


Figure B.2. Results for ART2 predicted by LSSVR.

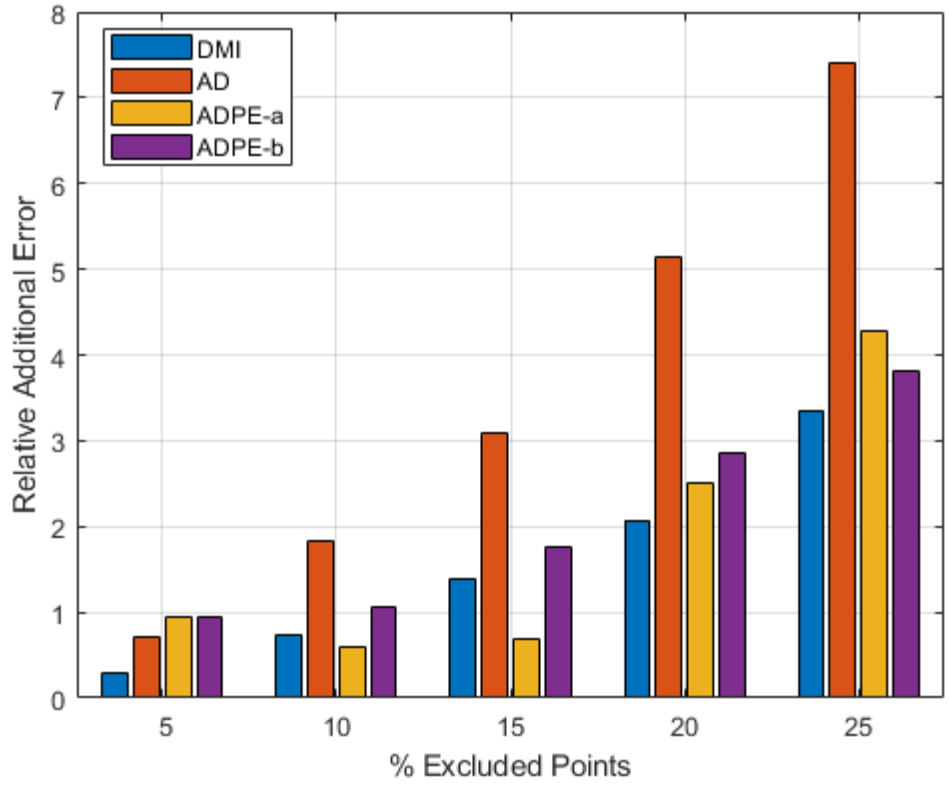


Figure B.3. Results for BNZ2 predicted by Lasso.

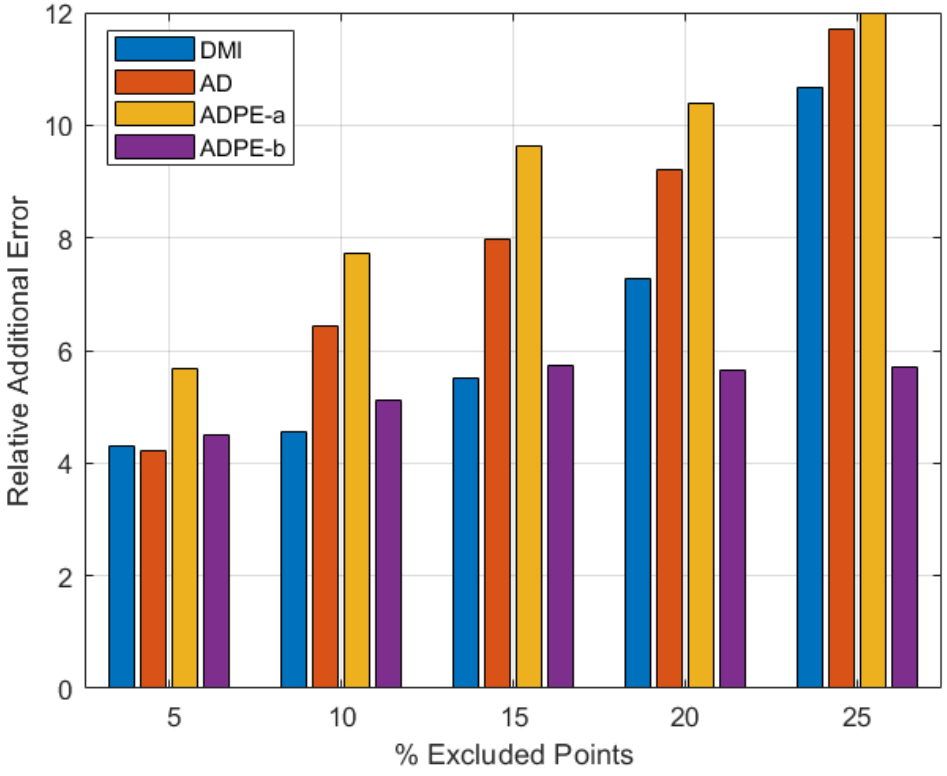


Figure B.4. Results for BNZ2 predicted by LSSVR.